DOI: 10.1111/cgf.13571
Pacific Graphics 2018
H. Fu, A. Ghosh, and J. Kopf
(Guest Editors)

*Volume 37 (2018), Number 7*

# Skeletex: Skeleton-texture Co-representation for Topology-driven Real-time Interchange and Manipulation of Surface Regions

M. Madaras[†1,2] and A. Riečický[1,2] and M. Mesároš[1,3] and M. Stuchlík[1] and M.Piovarči[4]

[1]Skeletex Research, Slovakia
[2]Department of Applied Informatics, Comenius University, Slovakia
[3]Masaryk University Brno, Czech Republic
[4]Università della Svizzera italiana, Switzerland

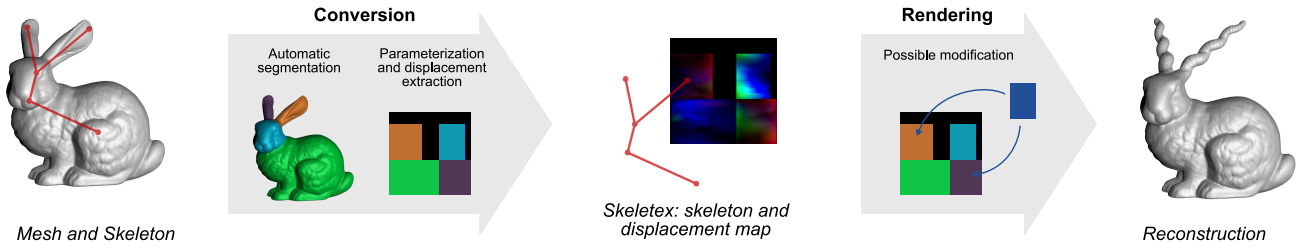


**Figure 1:** *The Skeletex encodes a 3D polygonal mesh as a skeleton and a vector displacement texture. The space around the skeleton is spatially subdivided and the subdivision is used for mesh segmentation. Each sub-space of skeletal spatial subdivision is deformed into a pose-invariant normalized bone tangent space. The skeleton-based parameterization is optimized in order to store a vector displacement map in the bone tangent space. A reconstruction of the model can be performed from the skeleton and the extracted texture map with possible surface modifications; the processing is done on the GPU in real-time.*

## Abstract

*Mesh processing algorithms depend on quick access to the local neighborhood, which requires costly memory queries. Moreover, even having access to the local neighborhood is not enough to efficiently perform many geometry processing algorithms in an automatic or semi-automatic way. As humans, we often imagine mesh editing at the level of topological information, e.g., altering surface features, adding limbs, etc., which is not supported by current data structures. These limitations come from the widely used mesh representations because the needed information is not implicitly defined by the structure. We propose a novel model representation called* Skeletex. *Each 3D model is decomposed into two elements: a skeletal structure that encodes the model topology and a vector displacement map to capture fine details of the geometry. Such a co-representation contains the topology information, as well as the information about the local vertex neighborhood at each texel. Additionally, our data structure facilitates an automatic skeleton-based cross-parameterization. This allows us to implement the mesh manipulation tasks in parallel, using a unified streamlined pipeline that directly maps to the GPU. We demonstrate the capabilities of our data structure by implementing surface region transfer and mesh morphing of 3D models.*

## CCS Concepts

•***Computing methodologies*** → ***Mesh models;***

## 1. Introduction

Mesh modeling and processing have a wide variety of applications in computer graphics, from creating realistic virtual characters in movies and games, to designing new models in industry and fabrication. Development of more advanced capturing and modeling

† madaras@skeletex.xyz

technologies allows us to create 3D meshes with an increasing level of detail [IH09,NFA*15]. Such an extreme number of vertices provides challenges for mesh processing algorithms that struggle to provide real-time feedback [BBVK04]. The lack of real-time computation inhibits manipulation of the objects. This results in a slower design process and inability to do procedural modifications on the fly, as artists and engineers have to wait for the computation to finish.

To accelerate the mesh processing, many researchers looked into leveraging the massive computational power of modern GPUs [Ngu07, XGH*11, OGWB12]. However, implementing mesh processing algorithms on a GPU is a challenging problem [SKNS15]. This is caused by the fact that mesh processing algorithms are formulated using the local neighborhood of a vertex. Therefore, efficient access to this information is essential to achieve good performance [YLPM05]. Commonly used data structures (e.g., half-edge, winged edge) were developed for CPU computation, and thus are not well suited for GPU hardware, where memory access is one of the most expensive operations [OLG*07]. A partial solution might be to store the mesh as a texture [GGH02] or as a matrix [ZSS17]. Moreover, a match between two models on a topological level is required for some mesh processing algorithms, e.g., detail transfer or morphing. Reconstructing such information from mesh data requires significant effort [BHS*17].

We propose a novel data structure for the representation of 3D models called *Skeletex*. Our idea is based on a common technique that splits a model representation into a low-polygonal mesh supported by a displacement texture [SKU08]. We take this idea to a higher level of abstraction, making the low-polygonal base mesh as decimated as possible, described only by a skeleton. We attach each skeleton segment to a displacement map that captures the surface geometry. The choice to use a skeleton and a vector displacement map works well with current GPU hardware and allows fast rendering of the data structure. By carefully constructing the parameterization domain of the texture and combining it with the skeleton's topology, we can ensure that the local neighborhood information is always accessible. This makes our representation suitable for mesh processing applications such as surface interchange and manipulation. Finally, by storing the geometry data inside a texture, we can reformulate selected mesh processing algorithms to work in texture space, eliminating their dependency on the number of vertices and making them suitable for GPU parallelization. To summarize, our main contributions are:

- description of our novel data structure based on skeleton-mesh co-representation that simplifies and unifies the mesh processing algorithms into a vertex-independent, texture-space and GPU-friendly implementation,
- an automatic method to convert input manifold meshes into our data structure, and
- a proposed solution for common skeleton-texture co-representation problems and pose-independent storage of surface details.

## 2. Related Work

Encoding the surface information into a texture space enables efficient GPU-powered mesh operations. However, to implement such operations in image space, the data structure has to have two properties. First, the structure should encode topological and neighborhood information correctly and consistently even on the texture seams. Second, we require the parameterization to be bijective and to preserve small-scale geometric details. In this section, the relevant works related to both requirements are discussed, as well as existing texture mesh representations.

In 3D asset creation applications, multiple representations are used to achieve maximal efficiency. For example, the artists prepare the 3D models using implicit representations with commercial software, e.g., ZBrush [PIX12] or Mudbox [Aut12]. Later, they export a model suitable for rendering: a low-polygon mesh enhanced with normal, displacement, or vector-displacement maps, which encode the high-frequency geometric details. However, this requires keeping two copies of the same model, as the rendering copy is hard to edit later on [TL17]. This limitation comes from the data representation. The high-frequency geometric details are represented in a texture map which lacks the topological information required for further mesh processing.

The topology of a 3D model can be represented by a graph structure commonly referred to as a skeleton. Skeletons are a well-known and often-used representation in computer graphics and computer vision. They are mostly used for shape description, shape analysis and shape matching [CDS*05, SSGD03, YTSG16, ATCO*10]. Moreover, the skeletons are used as a control structure for skeletal animation and as a structure for modeling and surface parameterization [JS11, JBK*12, JLW10, BMW12, Mv12, PSF04]. While the skeleton is a useful tool to represent the topology, it lacks high-frequency data needed to represent the detailed surface of a 3D model.

**Texture-Space Surface Encoding** The idea of encoding an entire high-detail 3D model into a single texture was proposed in *Geometry Images* [GGH02]. Buron et al. [BMGG15] demonstrated that such a representation is suitable for an on-mesh procedural generation. Carr et al. [CHCH06] and Feng et al. [FKY*10] extended the technique by dividing the surface into patches and generating a geometry image for each patch. Recently, Jang et al. [JH13] proposed an algorithm for generating displacement maps for extremely simplified base meshes. Liu et al. [LFJG17] proposed a resolution-independent seamless parameterization for mesh encoding. While these approaches enable encoding of surface details into texture, they do not store any topological information. Therefore, mesh operations requiring this information become very challenging, e.g., mesh morphing or topology-aware transfer of surface regions. In our work we solve this issue by designing a data structure that explicitly encodes the topology information at each location of an object's surface.

**Topology-Aware Parameterization** The topology information is a useful tool for parameterization as it allows dividing the parameterization problem into several smaller problems by splitting the model surface into a set of regions. One such example is to split the mesh into a Reeb graph [PSF04, ZMT05] that breaks the surface into several segments. Alternatively, the segmentation can be based on the volume of the geometry either with a regular grid or an octree [LD07]. Tarini et al. [Tar16] demonstrated how to effi-

ciently encode UV maps in a volume, thus enabling their application to run in real-time on the GPU. *PolyCube-Maps* [THCM04] propose to parameterize objects using a set of axis-aligned unit cubes that approximate the object volume. *Quad Layout* [ULP*15] pushes this idea even further by parameterizing an object by a set of skeleton-aligned quadrilaterals. The skeletal structure can also be used to transfer surface encoded attributes such as skinning weights [MPD*14] or segmentation [WYHC14]. Topology-aware parameterization techniques can be combined with state-of-the-art bijective [SS15, RPPSH17] or stretch minimizing [SSGH01, YBS04] parameterizations to preserve finer model details. However, such a combination is not straightforward. For example, it is not clear how to represent the geometry of the models to maintain consistent mapping during animation. On the other hand, Skeletex stores geometry details in a pose-agnostic manner. This allows direct support of mesh animation.

**Topology-Driven Mesh Representations** The usefulness of high-level topological information led researchers to incorporate it into their data structures. Bærentzen et al. proposed to represent 3D models as *Polar-Annular Meshes* (PAM) [BAS14]. PAM is a co-representation between a skeleton and a mesh that can be used for efficient editing and modeling. *Sphere-Meshes* [TGBE16], *B-Mesh* [JLW10], and ZBrush [PIX12] use implicit geometric primitives to encode the surface. Leblanc et al. [LHP11], Bærentzen et al. [BMW12], and Thiery et al. [TGBE16] employ a graph structure augmented with volume descriptors. Thiery et al. [TBTB12] propose to represent a model with topological cylinders. The topology of the model is captured by a curve skeleton and the geometry is represented by a set of generalized cylinders for each skeletal node. A similar data structure was later used by Yin et al. to reconstruct 3D models from scanned point clouds [YHZ*14]. Zhou et al. [ZYH*15] pushed the idea of generalized cylinders even further by finding an optimal decomposition of a 3D model into parts and showed applications such as morphing between two objects. Skeleton driven polycube representations were proposed by Liu et al. [LZLW15] and Livesu et al. [LMPS16]. These methods can align the polycubes with the skeletal structure to create a better match with the encoded surface. Bhagvat et al. [BJCW09] propose a GPU-friendly structure that captures the surface detail via conical frusta and uses relief mapping to reintroduce high-frequency details. The main disadvantage of these methods is that to represent fine details, e.g., spikes on a dragon's neck, an increasingly complex structure is required. Thus, time and memory complexity are much higher for highly detailed meshes. Moreover, each complex surface detail has to be associated with its own skeleton segment in order to be captured. This complicates the representation and degrades the intended topological correspondence. Thus, such representation cannot be used in combination with non-reliable curve-skeletons [CSM07]. A natural solution would be to augment the representations with a vector displacement map. However, this is not a trivial task. It comes with many complications, e.g. seam handling, and vector displacement map parameterization selection. In contrast, our data structure was designed from the ground up to properly handle seam boundaries and capture surface detail in an efficient manner, either by displacement or vector displacement maps.

## 3. Skeletex Representation

The Skeletex data structure consists of two components: a skeleton and a vector displacement map. In this section, we describe how to convert a 3D model represented by a triangular mesh into the structure. First, we need a skeleton of the model, which is used for an automatic mesh segmentation (Section 3.1). The skeleton also controls the texture layout subdivision into sub-textures, where each sub-texture is parameterized according to properties needed by the application (Section 3.2). In order to maintain the surface continuity between neighboring segments, bone segment neighbors are fixed in texture space based on the skeleton bind pose (Section 3.3). Surface compatibility between different models is achieved using normalized inter-space, called bone tangent space (Section 3.4). Finally, the Skeletex structure is directly reconstructed and rendered on the GPU (Section 3.6). The whole presented pipeline is depicted in Figure 1. The results and the analysis of the reconstruction process can be found in Section 4.

### 3.1. Skeleton and Mesh Segmentation

The Skeletex structure is based on a mapping between the mesh surface regions and bones of the skeleton, which is achieved using skeleton-guided mesh segmentation. In our experiments, we used manually generated skeletons provided by artists. Methods that provide automatically generated segmentation [ATC*08, SSCO08, BKR*16] could be used if they met certain criteria. First, we require that the neighboring segments are neighbors in the skeletal representation as well. Second, if the cutting area is not a plane, we need an interpolation function to encode our parameterization. For more details see Section 6. A skeleton is an oriented tree graph in which we denote the edges as bones. A skeleton bone $B$ is defined by its head and tail end nodes $H$ and $T$, where $H$ is the parent node and $T$ is the child node of the hierarchy. Each bone has an orthonormal basis of three vectors $(\vec{o},\vec{s},\vec{b})$, where $\vec{o} = \frac{T-H}{\|T-H\|}$, $\vec{s}$ is an arbitrary unit vector orthogonal to $\vec{o}$, and $\vec{b} = \vec{o} \times \vec{s}$. Vector $\vec{s}$ can be additionally modified by an artist for morphing applications, in order to enhance the visual fidelity of morph inter-states.

The segmentation process, and also the other processes, rely on the definition of *separation planes*. For each bone $B_i$ neighboring with bone $B$, there is a single unique separation plane $SP_i$. The separation plane $SP_i$ is defined by a common point of the two bones $B$ and $B_i$ and a normal vector constructed in such a way that the angles between the plane and lines defined by each of the bones are equal, as depicted in Figure 2, left. All planes constructed for a single bone $B$ and its neighbors form a plane set $\mathbb{P}_B$. An illustration of such a plane set in an example bone configuration can be found in Figure 2, right.

Each of the separation planes from $\mathbb{P}_B$ defines a closed half-space, where bone $B$ always lies in its positive partition, decided by an orientation of the plane normal. An intersection of these half-spaces forms a sub-space for a bone $B$, denoted as $Sub_B$. Therefore, to decide if an arbitrary point lies in $Sub_B$, it must lie in all half-spaces defined by planes from $\mathbb{P}_B$ as well. This way the $Sub_B$ segments the space around the bone in a deterministic way.

In order to obtain a surface region which corresponds to a bone $B$, the whole mesh is cut by all the planes from $\mathbb{P}_B$ as depicted in

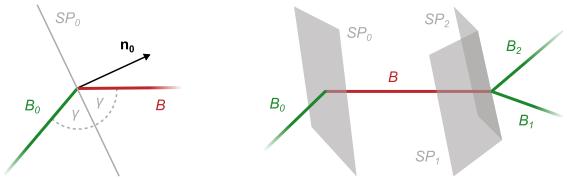


**Figure 2:** *Separation planes for bone B. Each pair of bones has a single unique separation plane SP defined by its normal vector, holding the same angle to both bones (left); an example of a bone configuration (linear on the left and branch on the right side of the bone B) with illustrated separation planes (right).*

**Figure 4:** *Visualization of mesh segmentation driven by the skeleton (left) and texture layout with regions assigned for each sub-mesh (right).*

Figure 3. The cutting procedure introduces new vertices over the edges of the mesh, which are stored in a boundary set $Bound_B$, as well as all the original vertices lying on either of the planes of $\mathbb{P}$. Afterwards, we filter the $Bound_B$ by discarding all vertices outside of $Sub_B$. To find a single component on the mesh, a flood-fill algorithm with restriction to the $Bound_B$ is performed over the mesh vertices. As an entry point, any vertex from $Sub_B$ can be used. In our case, we selected an arbitrary vertex from $Sub_B$, not present in $Bound_B$, lying in one-ring proximity to any vertex from $Bound_B$. Following the same procedure for each single bone of the skeleton, the final segmentation of the mesh is obtained.

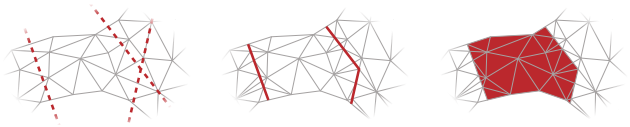

**Figure 3:** *The mesh is cut by separation planes (left); new vertices are introduced and the filtered boundary set is found (middle); flood-fill algorithm is performed to get the connected mesh component (right).*

### 3.2. Piece-wise Skeleton Segment Parameterization

We perform a skeleton-based subdivision of the texture space. First, a rectangular sub-texture is formed for each skeleton segment and packed into a single layout using a kd-tree (see Figure 4), similarly to [ULP*15, Mv12]. The size of the sub-texture is determined using a heuristic based on the skeleton segment length. Finally, each sub-texture region needs to be parameterized by a mapping $\mathbb{F} : (x,y,z) \mapsto (u,v)$. The parameterization approach to be used depends on the Skeletex properties needed by the application. Moreover, these properties might differ for each segment. We provide two different example cases in the paper. The coordinates $(u,v)$ are local coordinates for the specific rectangle in the texture; the sub-texture coordinates are defined in the interval $\langle 0,1 \rangle$.

#### 3.2.1. Separation Planes Driven Parameterization (SPD)

The parameterization of a surface point $P(x,y,z)$ is driven by a configuration of separation planes $\mathbb{P}_B$. A bone is denoted as a *root bone*, if it has neither parent nor sibling in the hierarchy. A bone is denoted as a *leaf bone*, if it has no child bones in the hierarchy. We distinguish two cases how the parameterization of $P(x,y,z)$ is calculated based on the parameter $p = \vec{o} \cdot (P - H)$. In the first case, the bone is a leaf and $p > \|T - H\|$, or the bone is a root and $p < 0$; in that case, the point $P$ belongs to a cap. Cap points are parameterized using a spherical coordinate system, for a single hemisphere. In the second case, a line $L_P$ is defined by a point $P$ and a directional vector which is parallel to the bone $B$ as depicted in Figure 5. The parallel line $L_P$ intersects all planes in $\mathbb{P}_B$. If the bone is a leaf or root, a plane perpendicular to the bone is added to the $\mathbb{P}_B$ on the tail or head side, respectively. We select two intersection points closest to $P$ in both directions $\pm\mathbf{d}$ and recover the segment points $A_H, A_T$. The coordinate $u$ is computed as $u = \frac{\|P - A_H\|}{\|A_T - A_H\|}$, and coordinate $v$ is computed as $v = \frac{\beta}{2\pi}$, where $\beta$ is the angle between the bone basis vector $\vec{s}$ and $\vec{s'}$. The vector $\vec{s'}$ is defined as $\vec{s'} = P - P'$, where a point $P' = H + p\vec{o}$. The *SPD* parameterization has uniform sampling along the bone but does not guarantee bijective mapping, as can be seen in Figure 6, bottom. The surface details that cannot be parameterized by the *SPD* in bijective way are referred in this paper as non-convex surface details.

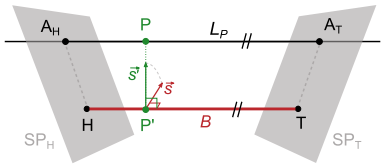


**Figure 5:** *Parameterization of a surface point P using a line $L_P$ that is parallel to the bone B.*

#### 3.2.2. Vertex Sampling Maximizing Parameterization (VSM)

For each segmented sub-mesh, the outer $(u,v)$ coordinates are fixed to the boundary of assigned regions, using uniform sampling. To preserve more complex surface details, the inner sub-mesh vertex coordinates can be further refined using an arbitrary local parameterization method. In our case, the texture coordinates are iteratively refined, similarly to Yoshizawa et al. [YBS04]. Each iteration consists of weighted barycentric mapping [Flo97]. However, the weight $w_{i,j}$ for an edge between vertices $i,j$ is computed as $w_{i,j}^{t+1} = w_{i,j}^{t} \cdot A_{i,j}^{t}$, where $w_{i,j}^{t}$ is the weight from the previous iteration and $A_{i,j}^{t}$ is the sum of the area of adjacent triangles with an edge

between vertices $i, j$ from the previous iteration. In the first iteration, simple unit weights $w^0_{i,j} = 1$ are used. Such weights balance the triangle area in texture space, which increases the probability of sampling every triangle at the cost of texture distortions. Unlike the *SPD*, the *VSM* parameterization guarantees bijective mapping. For a better view of *VSM* see Figure 6, top.

#### 3.2.3. Hybrid Parameterization

The *VSM* parameterization is more suitable to store the surface geometry since it preserves non-convex features. However, the parameterization induces significant stretch and tangential shifts into the texture space. On the other hand, *SPD* parameterization creates minimal tangential shift in the texture, but non-convex features cannot be preserved. Therefore, we also propose a segment-wise hybrid approach; the *VSM* is used for skeleton segments in which non-convex details need to be preserved, and *SPD* is used for segments where minimization of tangential shifts is desired (Figure 6, middle).

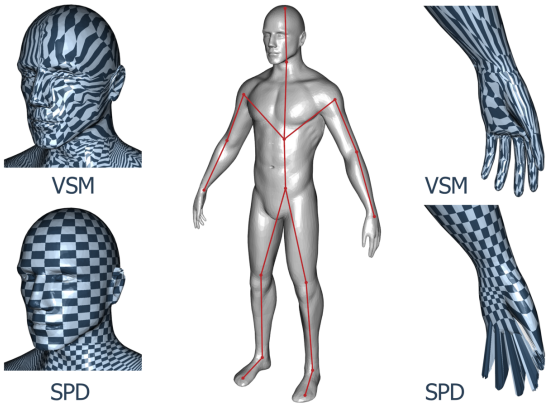


**Figure 6:** *Comparison of* VSM *parameterization (top) and* SPD *parameterization (bottom) on hand and head parts. The full model in the middle uses a hybrid approach:* VSM *for the hands and* SPD *for the rest of the body.*

### 3.3. Fixing Texture-space Segment Neighbors

Each neighboring bone influences a specific part of the bone surface. Therefore, the texture space needs to be split into a set of $v$ coordinate intervals $\mathbb{K}$ according to the neighboring configuration. The interval set $\mathbb{K}$ cannot change during skeleton pose change, in order to keep texture parts interchangeable for different poses.

For each head and tail node $N \in \{H, T\}$ of bone $B$ we collect all separation planes from $\mathbb{P}_B$ transecting the node $N$ and denote it as a set $\mathbb{P}_N$. Next, we perform pair-wise intersections of all planes in $\mathbb{P}_N$. Each intersection forms two rays transecting the point $N$ in directions $\pm\vec{r}$ (see Figure 8, left). We filter the rays and accept only rays such that $\{\mathbf{r} \mid \forall P \in \mathbb{P}_N : \mathbf{n_P} \cdot \mathbf{r} \geq 0\}$, where $\vec{n_P}$ is a normal of plane $P$. For the remaining rays we calculate the $v$ coordinates, according to the *SPD* parameterization (Section 3.2.1), giving interval bounds $I_i$ (Figure 8, left) used for calculation of the roll quaternion (Section 3.4.1). All interval bounds split the texture space into interval set $\mathbb{K}$ (Figure 8, right), computed in bind pose.

### 3.4. Bone Tangent Space

In order to maintain compatibility between the displacement maps of two models in different poses, the geometry is stored in the *bone tangent space* (see Figure 7), representing a pose-independent, normalized form of the surface. In order to create the bone tangent space, two successive transformations $\Phi_B$, $\Psi_B$ need to be performed. The first transformation $\Phi_B : (u, v) \mapsto q$ deforms $Sub_B$ according to the quaternion $q$. As a result, a uniform tube is obtained. The second transformation $\Psi_B : (u, v) \mapsto q'$ consists of two rotations. The first is the rotation around the axis $\vec{o}$ by the angle $-2\pi v$ that unrolls the uniform roll into a plane, and the second rotation transforms the base of the plane from world space to bone space $(\vec{o}, \vec{s}, \vec{b})$. Consequently, in the reconstruction phase, the geometry can be unpacked from the bone tangent space using $\Psi_B^{-1}$ and $\Phi_B^{-1}$.

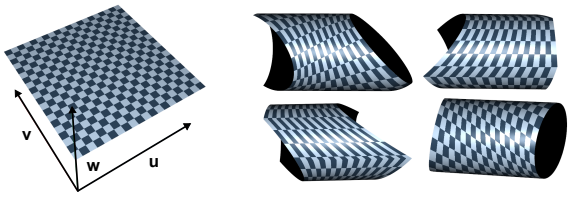


**Figure 7:** *A plane $w = 0.5$ defined in bone tangent space (left) describes various surface deformations depending on pose configuration (right).*

The quaternion $q$ can be split into two rotations, a roll quaternion $q_r$ (where the rotation axis is vector $\vec{o}$) and a yaw-pitch quaternion $q_{yp}$ (where the rotation axis is vector $\vec{p} = \vec{o} \times \vec{s'}$, with $\vec{s'}$ being $\vec{s}$ rotated around axis $\vec{o}$ by the angle $2\pi v$). Therefore, the roll quaternion $q_r$ and the yaw-pitch quaternion $q_{yp}$ are calculated first, and then the final quaternion is computed as $q = q_{yp} q_r$. The quaternion $q$ for the bone cap is an identity.

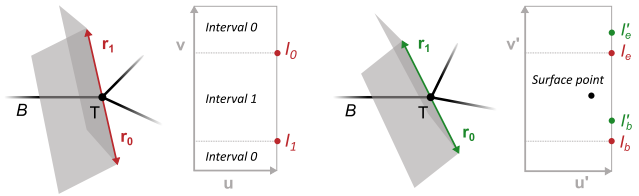


**Figure 8:** *In the bind pose, two rays $r_0, r_1$ are generated by the intersection of separation planes. The rays split the $v'$-coordinate space into two intervals separated by bounds $I_i$ (left). In the deformed skeleton pose, the corresponding rays $r_0, r_1$ and intervals change. The $\alpha_H, \alpha_T$ angles are computed from differences in $v'$ coordinates of interval bounds $I_b, I_e$ and $I'_b, I'_e$ for each surface point lying in a specific interval (right).*

#### 3.4.1. Roll Quaternion

The roll quaternion consists of two secondary roll rotations $q_{rN}$ defined separately for the head and the tail nodes $N \in \{H, T\}$. In the case of a linear node $N$ where the bone $B$ has only a single neighbor $B_i$, the secondary roll rotation is defined as a rotation with reference axis $\vec{o}$ and the rotation angle equals the angle between $\vec{s}$ and $\vec{s'_i}$,

where $\vec{s}'_i$ is the neighbor basis vector $\vec{s}_i$ rotated around the vector $\vec{c} = \vec{o} \times \vec{o}_i$, such that $\vec{s}'_i \perp \vec{o}$. For a branch node, the secondary roll is calculated for each $v$ separately. Assume an interval in the bind pose $I = \langle I_b, I_e \rangle \mod 1$, $I \in \mathbb{K}$ (Figure 8, left), where $v \in I$, and the corresponding interval in the current pose $I' = \langle I'_b, I'_e \rangle \mod 1$. First, parameter $v'$ is found as $v' = (I'_b + \frac{v - I_b}{|I|} |I'|) \mod 1$, and an angle $\alpha_N$ is computed according to $\alpha_N = 2\pi(v' - v)$. Finally, rotations $q_{rN}$ given by angles $\alpha_N$ from both nodes and the reference axis $\vec{o}$ are interpolated according to the parameter $u$, resulting in the final roll quaternion $q_r$.

### 3.4.2. Yaw-Pitch Quaternion

The following is performed for both end nodes $N \in \{H, T\}$. First, the bone basis vector $\vec{s}$ is rotated around axis $\vec{o}$ according to secondary roll quaternion $q_{rN}$, resulting in the vector $\vec{x}_N$. For the end node $N$, a point $M_N = N + \vec{x}_N$ is calculated. Next, line $L_{M_N}$ given by the vector $\vec{o}$ and the point $M_N$ is constructed. Point $J_N$ is obtained as the intersection of $L_{M_N}$ and $z$, where $z$ is the separation plane for the interval $I$ (Section 3.4.1). Then vector $\vec{f}_N = J_N - N$ is rotated around the axis $\vec{o}$ by the same angle as the angle between $\vec{x}_N$ and $\vec{s}''$, where $\vec{s}''$ is $\vec{s}$ rotated by quaternion $q_r$. Finally, vectors $\vec{f}_N$ for both nodes are interpolated according to the parameter $u'$, resulting in the vector $\vec{f}'$. The final yaw-pitch quaternion $q_{yp}$ is obtained as rotation from the vector $\vec{x}_N$ and $\vec{f}'$.

## 3.5. Cross-parameterization Roll Correction

The Skeletex structure can be directly used to map a segment from source skeleton $S_s$ to any other segment of destination skeleton $S_d$, when skeletons $S_s$ and $S_d$ have the same neighbor intervals in the texture space. In order to support cross-parameterization between arbitrary models $M_d$, $M_s$ with different skeleton poses $S_d$, $S_s$, the parameterization of the model $M_s$ needs to be corrected. The correction consists of shifting the coordinate $v'$ by $\alpha$ according to the *roll quaternion* explained in Section 3.4.1, where the *bind pose* is $S_s$ and the *current pose* is $S_d$.

## 3.6. Reconstruction and Rendering

The reconstruction of the detail mesh is achieved by displacing the base mesh geometry. Before the displacement, the vectors must be transformed from the bone tangent space back to the world space. In order to reconstruct the mesh model back from the Skeletex, for each sub-texture region, a regular quad lattice with the size of the region is generated on the CPU; each grid vertex corresponds to exactly one texel (Figure 9). The grid is uploaded to the GPU and it is used as a base mesh that is displaced in the shader program. The corresponding displacement vector for each vertex is first rolled back and transformed using the inverse transformations $\Psi_B^{-1}\Phi_B^{-1}$, respectively. Second, its distance from the bone is modified to be the same as in the original model (Section 3.6.1). The whole process of the reconstruction is done in real-time on the GPU.

### 3.6.1. Restoring Original Distance from Bones

With the displacement vectors $T(u,v)$ and quaternion-based deformations $Q(u,v)$ (described in Section 3.4), the distance of the surface from the bone has a tendency to change as a pose differs from the bind pose, because the displacements are rotated (as shown on the left side of Figure 10). We restore the bind pose volume of a reconstructed surface by preserving its distance from the bone, disregarding the pose. We encode the distance between a surface point and the bone segment as the length of a displacement vector, during creation of the displacement map.

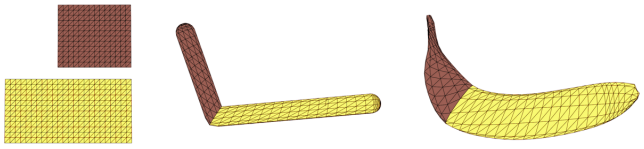

**Figure 9:** *Reconstruction of the Skeletex. From left to right: a generated grid for each texel, reconstruction using a constant displacement vector, final displaced surface.*

Distance-preserving reconstruction (see Appendix A for details) considers two cases:

1. Distance is restored perpendicularly to the bone segment.
2. Distance is restored according to one of the bone end nodes $N$.

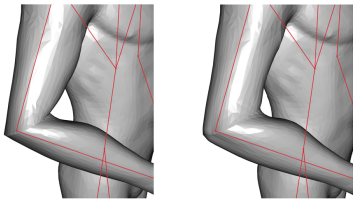

**Figure 10:** *Without the distance preservation (left), the reconstructed mesh thickness may change. The distance from the bone has to be stored in the texture (right) in order to preserve the original local thickness of the mesh.*

## 4. Results

In this section, we present a technical evaluation of the Skeletex data structure. The quality of the reconstruction of different models is evaluated. We start by exploring the impact of texture size on the quality of our reconstruction (Section 4.1). Next, we present results of our reconstruction of various geometries with different skeletons (Section 4.2). The results of skeleton-based segmentation are visualized in Figure 11 and Figure 12. The evaluation of the time needed for the preprocessing stage of conversion to the Skeletex structure can be seen in Table 1 and the final rendering (reconstruction, morphing, transfer) can be performed in real-time; we achieved approximately 90 fps on a GeForce GTX 1080 with the texture resolution $2048 \times 2048$ using a naive implementation.

## 4.1. Analysis of Reconstruction

After the input model has been converted into the Skeletex structure, the reconstruction of the original mesh can be stated as a remeshing problem. The quality of the new mesh directly depends on the texture resolution. To evaluate the impact of resolution, we

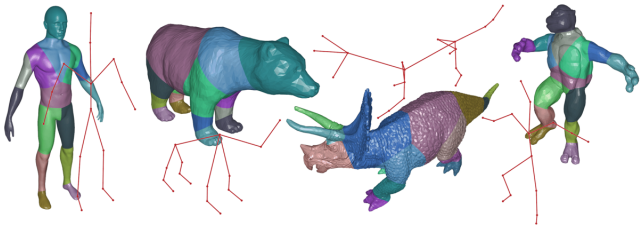

**Figure 11:** *Results of our skeleton-based segmentation on a set of example models.*

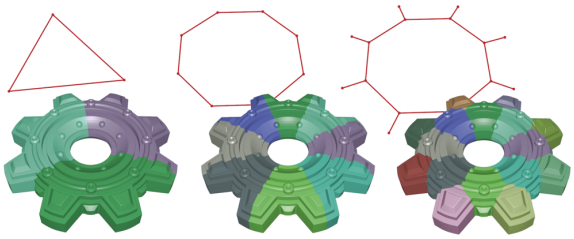

**Figure 12:** *Results of our segmentation and reconstructions of the same geometry using several cyclic skeletons given by an artist.*

reconstructed a set of models with varying texture size. For each model, we compute the metro distance [CRS98] between meshes normalized by the diagonal of the model's bounding box. The mean and standard deviation of the per-vertex distances for each model is visualized in Figure 14. From the data, we can see there is a significant gain up to a texture size of $512 \times 512$. Therefore, we selected this texture size for generating results in the remainder of this paper.

Our method seeks to preserve shape even for small texture sizes. Therefore, it is possible to use down-scaled textures to render our models with different levels of detail. Figure 13 shows how such a dynamic level of detail would look using the Einstein bust. Even

**Figure 13:** *The reconstruction of the Einstein model with different levels of detail; vector displacement maps with the resolution of* 32, 64, 128, 256, 512 *and* 1024. *The vertices in base triangulation are generated for every texel of the texture.*

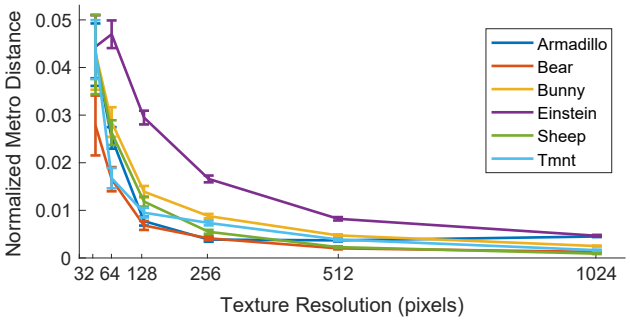


**Figure 14:** *An evaluation of reconstruction error with respect to texture resolution. The plot shows the mean and standard deviation of the metro distance normalized by the bounding box of each model.*

| Model | # bones | # vertices | $t_{OM}$ | $t_S$ | $t_P$ |
|---|---|---|---|---|---|
| Bear | 15 | 5k | 3455 | 2884 | 143 |
| Turtle | 13 | 7k | 4038 | 3363 | 169 |
| Sheep | 15 | 15k | 18373 | 13979 | 366 |
| Armadillo | 17 | 17k | 30685 | 25360 | 450 |
| Einstein | 2 | 24k | 3676 | 10815 | 525 |
| Bunny | 4 | 35k | 15797 | 6588 | 841 |

**Table 1:** *The evaluation of time complexity needed for the preprocessing stage of Skeletex creation. The texture of size* $512 \times 512$ *is used for all the models in the table. The* $t_{OM}$, $t_S$, $t_P$ *state time in ms needed for conversion into half-edge in OpenMesh [BSBK02], segmentation by separating planes and parameterization optimization, respectively.*

at a resolution of $32 \times 32$ pixels, the reconstructed model resembles a head. From $128 \times 128$ pixels the main facial features become visible, and they become prominent at a resolution of $256 \times 256$. Finally, a reconstruction without visible artifacts is achieved at a resolution of $1024 \times 1024$.

### 4.2. Reconstruction Results and Skeletex Evaluation

To evaluate our method we store various models in the Skeletex structure and reconstruct them on the GPU (Figure 15). For each model, we visualize its skeleton, displacement map, reconstruction, and error from the original mesh. The error is expressed as metro distance normalized by the diagonal of the model's bounding box. All the models are reconstructed with texture resolution $512 \times 512$. We can see that in general, our reconstruction achieves very small errors.

## 5. Example Applications

In this section, we present the results of the previously mentioned application of our Skeletex representation. Skeletex provides a unified framework for many algorithms and a natural way of porting the algorithms to the GPU. We demonstrate this by implementing two example applications.

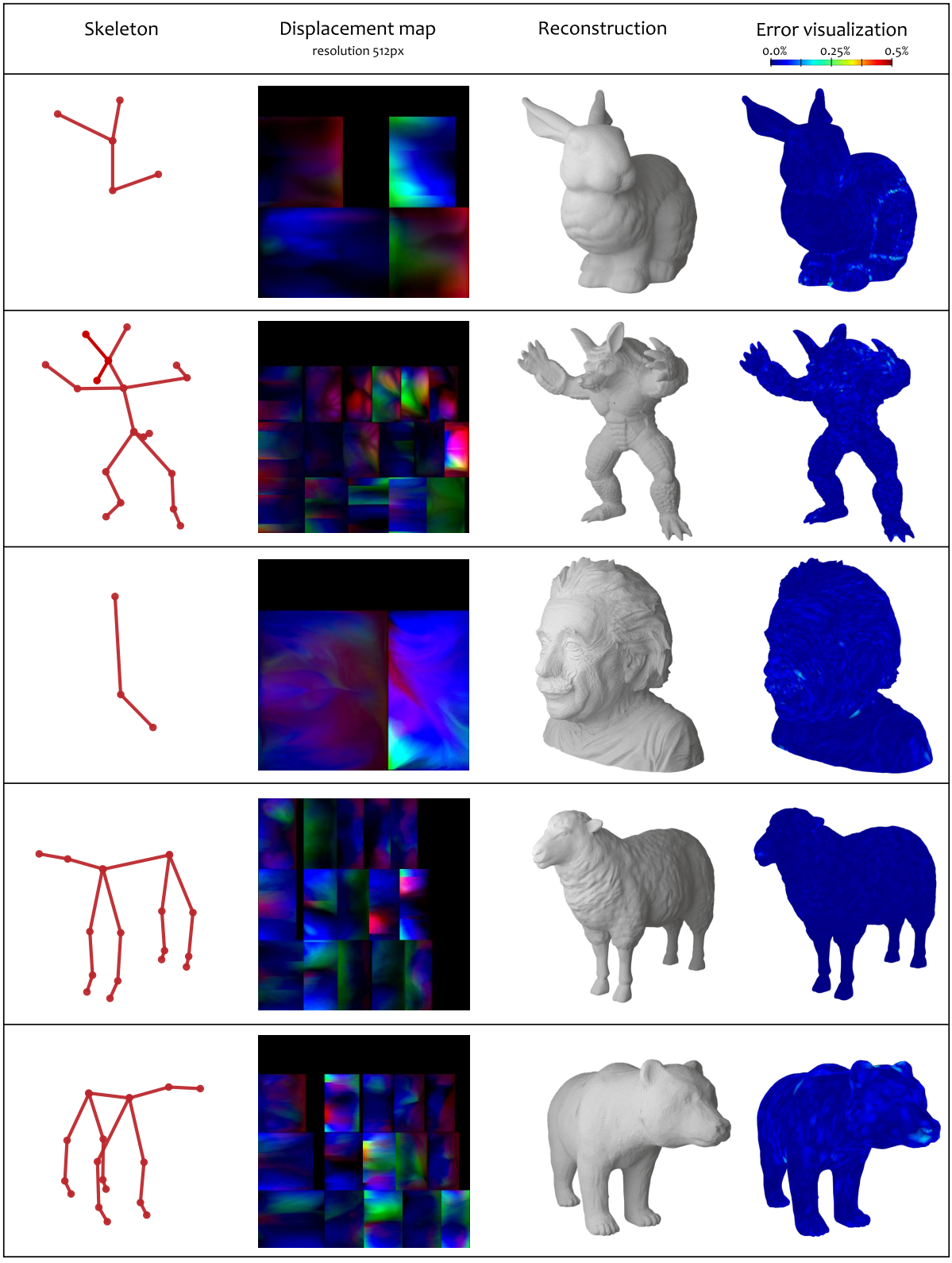


**Figure 15:** *From an input model converted into the Skeletex structure, the triangular mesh can be reconstructed. From left to right: skeleton of the input mesh model, parameterization domain of the mesh and the vector displacement map, reconstructed mesh with applied displacements, and the Hausdorff distance between the original model and the Skeletex reconstruction.*

**Texture-space Global Morphing** We present the mesh morphing application based on the interpolation of geometry using Skeletex. Our algorithm works as follows. In the first step, bone correspondences between the models are established. Skeletons with the same node branching are required. Once the correspondences are established, we can morph each segment of the mesh individually. This is done by linearly interpolating between the texture patches corresponding to matched segments. No further vertex correspondence search, nor other processing steps are necessary. The skeleton transformations are linearly interpolated as well.

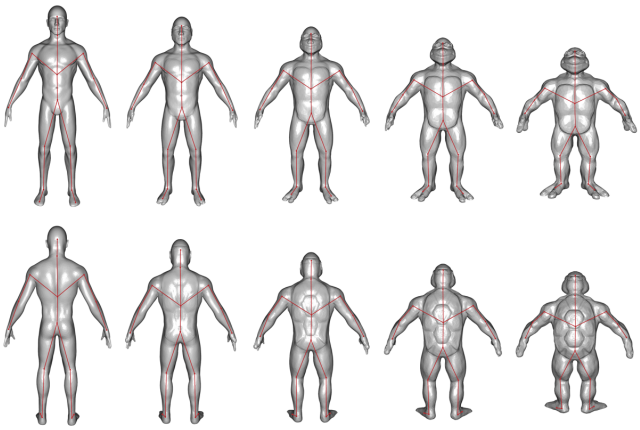

**Figure 16:** *Morphing of the human to the ninja turtle using a vector displacement map and* SPD *parameterization.*

When compared to parameterization-based morphing, [FJFS05], our approach facilitates the setup by removing manual steps, such as searching for polygon correspondence or parameter tweaking. Furthermore, no registration steps need to be performed; this is solved automatically by the representation itself and the roll correction. While the quality of our morphing is lower than the one of the offline implementation, e.g., [BHLW12], the performance is significantly higher, providing fully automatic real-time morphing. Thanks to our data structure, even a naive linear interpolation algorithm, used for the results in Figure 16, can achieve good results. These may be further improved by more sophisticated interpolation methods for image morphing [LLN*14].

**Surface Region Transfer** Having two models stored in our Skeletex structure, we can transfer parts of one model onto another. Both inner and leaf segments of the skeleton can be interchanged; however, the branching of the segments needs to be the same. This can be done by simply replacing the corresponding texture patch; however, the continuity of the mesh on the seam will not be guaranteed (Figure 19, right). We solve this problem by applying a blending function close to the patch boundary, as depicted in Figure 17. Thanks to explicit orientation information provided by our data structure, we can easily align the patches and create a seamless model. We performed such a transfer operation on the armadillo and bunny models using a cosine blending function (Figure 18).

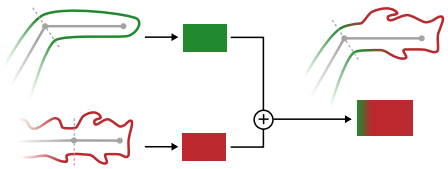

**Figure 17:** *Interpolation of transferred regions near the boundary. The displacements between regions are interpolated based on a function.*

**Figure 18:** *Transfer of surface regions of two models. The bunny's ears are replaced by horns (left), the armadillo's front paw is replaced by a pincer (right).*

## 6. Discussion

The proposed Skeletex structure enables converting triangular 2D manifold meshes to a cross-parameterized skeleton and texture map. The texture map is created in a predefined manner, guaranteeing access to the local neighborhood at each point. Furthermore, the texture contains alignment information which allows us to distinguish directions along and across the bone. This additional information stored in texture was leveraged in our application cases. Reproducing our application cases using *Geometry Images*, one would need to employ manual point registration and segmentation due to lack of topological data. On the other hand, when compared to techniques such as *PolyCube-Maps* or *Quad Layout*, our local optimization approach enables us to encode complex objects with varying feature size without additional overhead from progressively more complicated data structures.

The method does not have a general parameterization satisfying both the static reconstruction with non-convex parts and morphing. However, Skeletex is agnostic to the parameterization used to encode the surface details. The only condition is that we require rectangular parameterization boundaries to ensure easy interchange of bone regions. We opted for our parameterizations due to their simplicity and good resulting quality. An exploration of mapping strategies for surface detail encoding is an interesting area for future work.

In terms of memory efficiency, our data structure has relatively high space consumption. First, the texture-space encoding of the geometry usually needs more space than a standard polygonal mesh representation. Skeletex maintains the surface neighborhood connectivity in the texture space at the expense of texture memory consumption. Second, there are visible blank patches in the textures (Figure 15, displacement map). This happens because our optimization prefers to create longer rectangular patches. We could further optimize the space usage by choosing the texture size ap-

propriately. Finally, our textures store full float vector displacement maps, which are costly in terms of space.

### 6.1. Limitations

The conversion of a 3D model into the Skeletex structure requires a 2D-manifold mesh and an input skeleton. All nodes of the skeleton must be inside the mesh, and none of the bones should intersect the mesh. The main limitation of segmentation by separation planes is that only a mesh surface between the separation planes can be correctly segmented and parameterized. That means that all volumes behind the intersection of two planes from the opposite nodes of a bone cannot be represented by a Skeletex structure. You can see an example of this limitation in Figure 19, upper left. Also, skeletons have to be selected carefully, such that after the segmentation by separation planes there are no neighboring mesh segments which are not also neighbors topologically at the skeleton level (see Figure 19, lower left). If such an unsupported configuration occurs, the skeleton has to be modified by the artist. Finding an automated method is an aim for future work.

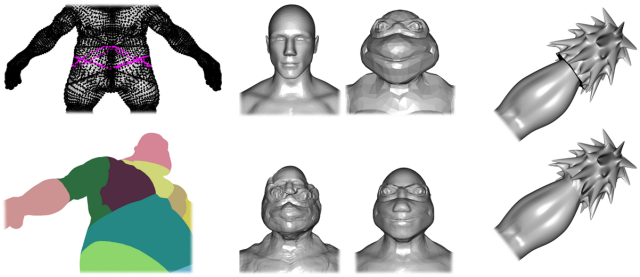

**Figure 19:** *An example of invalid segmentation (left), and a comparison of morphing using different parameterizations (middle); original models to be morphed on the top and morph results using* VSM *and* SPD *parameterizations on the bottom. Detail transfer without and with interpolation around boundary (right).*

The quality of our morphing algorithm depends on the parameterization. The *VSM* parameterization is suitable for static meshes; however, morphing and surface transfer creates undesired surface deformations (see Figure 19). On the other hand, *SPD* parameterization works well for morphing and transfer applications, but cannot capture non-convex surface details.

During surface region transfer we are forced into a trade-off between preserving details on the seam and visually continuous results. If we were to naively transfer incompatible geometries, the resulting surface would contain a visible seam line (Figure 19, right top). On the other hand, by applying our blurring strategy, we cannot preserve details on the texture seams (Figure 19, right bottom). If the transferred regions were designed to be interchangeable, we could turn off our smoothing strategy and produce results that have no visible seam line and preserve details on the seam.

The Skeletex animation can be easily performed by skeleton pose transformation; however, the visual quality of the animation depends on two main factors: first, the relative thickness of the mesh and second, the magnitude of the transformation. The thicker the mesh is, the smaller the range of motion the animation has (see Figure 20). The lighting artifacts visible on segmentation boundaries are characteristic for $C_0$ continuity. If any rotation exceeds the angle $\pi$, the interpolation of quaternions becomes ambiguous.

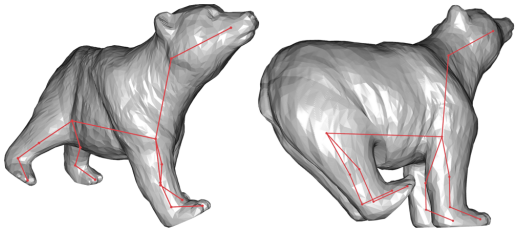

**Figure 20:** *The Skeletex structure does support animation when the skeleton is transformed (left). However, an extreme rotation of bones looks unnatural (right).*

## 7. Conclusions

We presented a new representation of 3D models called Skeletex. Our technique stores the surface of a 3D model as a skeleton and a vector displacement map. The primary contribution of our work is the analysis of the issues that stem from combining a topology-driven parameterization with a vector displacement map: seam handling, displacement map selection, and pose independent detail storage which guarantees compatibility of models in different poses. In contrast to other related approaches, Skeletex is capable of encoding fine surface details by a skeleton without degrading the topological correspondence. This enables image-space implementation of many mesh processing algorithms in a GPU-friendly format. We demonstrated the effectiveness of our structure on applications such as surface region transfer and mesh morphing. We believe that many more applications can benefit from texture-space representation of models supported with topological information.

## 8. Acknowledgments

A set of free 3D models used for evaluation was obtained from *www.thingiverse.com*, namely: Bear by YahooJAPAN published on November 12, 2013; Sheep by YahooJAPAN published on November 12, 2013; Albert Einstein Bust by LSMiniatures published on August 13, 2015; Teenage Mutant Ninja Turtle by Misfit410 published on May 10, 2017; COG - Medal #1 by Chrizz published on October 16, 2016; Triceratops by Mundocani published on February 15, 2014 and Crab by YahooJAPAN published on November 12, 2013. Ogre by Gabrielvfx published on October 10, 2011 and Male Human by Nixor were obtained from *www.free3d.com*.

## References

[ATC*08] Au O. K.-C., Tai C.-L., Chu H.-K., Cohen-Or D., Lee T.-Y.: Skeleton extraction by mesh contraction. In *SIGGRAPH '08: ACM SIGGRAPH 2008 papers* (New York, NY, USA, 2008), ACM, pp. 1–10. 3

[ATCO*10] Au O. K.-C., Tai C.-L., Cohen-Or D., Zheng Y., Fu H.: Electors voting for fast automatic shape correspondence. In *Computer Graphics Forum (In Proc. of Eurographics)* (2010), vol. 29. 2

[Aut12] AUTODESK: Mudbox, 2012. http://www.autodesk.com/. 2

[BAS14] BÆRENTZEN J. A., ABDRASHITOV R., SINGH K.: Interactive shape modeling using a skeleton-mesh co-representation. *ACM Transactions on Graphics (proceedings of ACM SIGGRAPH) 33*, 4 (2014). 3

[BBVK04] BOTSCH M., BOMMES D., VOGEL C., KOBBELT L.: Gpu-based tolerance volumes for mesh processing. In *Proceedings of the Computer Graphics and Applications, 12th Pacific Conference* (Washington, DC, USA, 2004), PG '04, IEEE Computer Society, pp. 237–243. 2

[BHLW12] BOJSEN-HANSEN M., LI H., WOJTAN C.: Tracking surfaces with evolving topology. *ACM Trans. Graph. 31*, 4 (July 2012), 53:1–53:10. 9

[BHS*17] BERKITEN S., HALBER M., SOLOMON J., MA C., LI H., RUSINKIEWICZ S.: Learning detail transfer based on geometric features. *Comput. Graph. Forum 36*, 2 (May 2017), 361–373. 2

[BJCW09] BHAGVAT D., JESCHKE S., CLINE D., WONKA P.: Gpu rendering of relief mapped conical frusta. *Computer Graphics Forum 8*, 28 (2009), 2131–2139. 3

[BKR*16] BALDACCI A., KAMENICKÝ R., RIEČICKÝ A., CIGNONI P., ĎURIKOVIČ R., SCOPIGNO R., MADARAS M.: GPU-based approaches for shape diameter function computation and its applications focused on skeleton extraction. *Comput. Graph. 59*, C (Oct. 2016), 151–159. 3

[BMGG15] BURON C., MARVIE J.-E., GUENNEBAUD G., GRANIER X.: Dynamic on-mesh procedural generation. In *Proceedings of the 41st Graphics Interface Conference* (Toronto, Ont., Canada, Canada, 2015), GI '15, Canadian Information Processing Society, pp. 17–24. 2

[BMW12] BÆRENTZEN J., MISZTAL M., WEŁNICKA K.: Converting skeletal structures to quad dominant meshes. *Computers & Graphics 36*, 5 (2012), 555 – 561. Shape Modeling International (SMI) Conference 2012. 2, 3

[BSBK02] BOTSCH M., STEINBERG S., BISCHOFF S., KOBBELT L.: Openmesh - a generic and efficient polygon mesh data structure, 2002. 7

[CDS*05] CORNEA N. D., DEMIRCI M. F., SILVER D., SHOKOUFANDEH A., DICKINSON S. J., KANTOR P. B.: 3D object retrieval using many-to-many matching of curve skeletons. In *Proceedings of the International Conference on Shape Modeling and Applications 2005* (Washington, DC, USA, 2005), SMI '05, IEEE Computer Society, pp. 368–373. 2

[CHCH06] CARR N. A., HOBEROCK J., CRANE K., HART J. C.: Rectangular multi-chart geometry images. In *Proceedings of the Fourth Eurographics Symposium on Geometry Processing* (Aire-la-Ville, Switzerland, Switzerland, 2006), SGP '06, Eurographics Association, pp. 181–190. 2

[CRS98] CIGNONI P., ROCCHINI C., SCOPIGNO R.: Metro: Measuring error on simplified surfaces. *Computer Graphics Forum 17*, 2 (1998), 167–174. 7

[CSM07] CORNEA N. D., SILVER D., MIN P.: Curve-skeleton properties, applications, and algorithms. *IEEE Transactions on Visualization and Computer Graphics 13*, 3 (2007), 530–548. 3

[FJFS05] FAN Z., JIN X., FENG J., SUN H.: Mesh morphing using polycube-based cross-parameterization: Animating geometrical models. *Comput. Animat. Virtual Worlds 16* (July 2005), 499–508. 9

[FKY*10] FENG W.-W., KIM B.-U., YU Y., PENG L., HART J.: Feature-preserving triangular geometry images for level-of-detail representation of static and skinned meshes. *ACM Trans. Graph. 29* (April 2010), 11:1–11:13. 2

[Flo97] FLOATER M. S.: Parametrization and smooth approximation of surface triangulations. *Comput. Aided Geom. Des. 14*, 3 (Apr. 1997), 231–250. 4

[GGH02] GU X., GORTLER S. J., HOPPE H.: Geometry images. *ACM Trans. Graph. 21* (July 2002), 355–361. 2

[IH09] IHRKE I., HEIDRICH W.: Acquisition of optically complex objects and phenomena. In *ACM SIGGRAPH 2009 Courses* (New York, NY, USA, 2009), SIGGRAPH '09, ACM, pp. 1:1–1:158. 2

[JBK*12] JACOBSON A., BARAN I., KAVAN L., POPOVIĆ J., SORKINE O.: Fast automatic skinning transformations. *ACM Transactions on Graphics (proceedings of ACM SIGGRAPH) 31*, 4 (2012), 77:1–77:10. 2

[JH13] JANG H., HAN J.: GPU-optimized indirect scalar displacement mapping. *Computer-Aided Design 45*, 2 (2013), 517 – 522. Solid and Physical Modeling 2012. 2

[JLW10] JI Z., LIU L., WANG Y.: B-Mesh: A modeling system for base meshes of 3D articulated shapes. *Computer Graphics Forum 29*, 7 (2010), 2169–2177. 2, 3

[JS11] JACOBSON A., SORKINE O.: Stretchable and twistable bones for skeletal shape deformation. *ACM Transactions on Graphics (proceedings of ACM SIGGRAPH ASIA) 30*, 6 (2011), 165:1–165:8. 2

[LD07] LEFEBVRE S., DACHSBACHER C.: TileTrees. In *Proceedings of the 2007 Symposium on Interactive 3D Graphics and Games* (New York, NY, USA, 2007), I3D '07, ACM, pp. 25–31. 2

[LFJG17] LIU S., FERGUSON Z., JACOBSON A., GINGOLD Y.: Seamless: Seam erasure and seam-aware decoupling of shape from mesh resolution. *ACM Trans. Graph. 36*, 6 (Nov. 2017), 216:1–216:15. 2

[LHP11] LEBLANC L., HOULE J., POULIN P.: Modeling with blocks. *The Visual Computer (Proc. Computer Graphics International 2011) 27*, 6-8 (June 2011), 555–563. 3

[LLN*14] LIAO J., LIMA R. S., NEHAB D., HOPPE H., SANDER P. V., YU J.: Automating image morphing using structural similarity on a halfway domain. *ACM Trans. Graph. 33*, 5 (Sept. 2014), 168:1–168:12. 9

[LMPS16] LIVESU M., MUNTONI A., PUPPO E., SCATENI R.: Skeleton-driven adaptive hexahedral meshing of tubular shapes. *Computer Graphics Forum 35*, 7 (2016), 237–246. 3

[LZLW15] LIU L., ZHANG Y., LIU Y., WANG W.: Feature-preserving t-mesh construction using skeleton-based polycubes. *Computer-Aided Design 58* (2015), 162 – 172. Solid and Physical Modeling 2014. 3

[MPD*14] MADARAS M., PIOVARČI M., DADOVÁ J. B., FRANTA R., KOVAČOVSKÝ T.: Skeleton-based matching for animation transfer and joint detection. In *Proceedings of the 30th Spring Conference on Computer Graphics* (New York, NY, USA, 2014), SCCG '14, ACM, pp. 91–98. 3

[Mv12] MADARAS M., ĎURIKOVIČ R.: Skeleton texture mapping. In *Proceedings of the 28th Spring Conference on Computer Graphics* (New York, NY, USA, 2012), SCCG '12, ACM, pp. 121–127. 2, 4

[NFA*15] NAGANO K., FYFFE G., ALEXANDER O., BARBIČ J., LI H., GHOSH A., DEBEVEC P.: Skin microstructure deformation with displacement map convolution. *ACM Trans. Graph. 34*, 4 (July 2015), 109:1–109:10. 2

[Ngu07] NGUYEN H.: *Gpu Gems 3*, first ed. Addison-Wesley Professional, 2007. 2

[OGWB12] OLANO M., GRIFFIN W., WANG Y., BERRIOS D.: Real-time gpu surface curvature estimation on deforming meshes and volumetric data sets. *IEEE Transactions on Visualization Computer Graphics 18* (10 2012), 1603–1613. 2

[OLG*07] OWENS J. D., LUEBKE D., GOVINDARAJU N., HARRIS M., KRÃIJGER J., LEFOHN A. E., PURCELL T. J.: A survey of general-purpose computation on graphics hardware. *Computer Graphics Forum 26*, 1 (2007), 80–113. 2

[PIX12] PIXOLOGIC: ZBrush, 2012. http://www.pixologic.com/. 2, 3

[PSF04] PATANÉ G., SPAGNUOLO M., FALCIDIENO B.: Para-Graph: Graph-based parameterization of triangle meshes with arbitrary genus. *Comput. Graph. Forum* (2004), 783–797. 2

[RPPSH17] RABINOVICH M., PORANNE R., PANOZZO D., SORKINE-HORNUNG O.: Scalable locally injective mappings. *ACM Transactions on Graphics 36*, 2 (Apr. 2017), 16:1–16:16. 3

[SKNS15] SCHÄFER H., KEINERT B., NIESSNER M., STAMMINGER M.: Local painting and deformation of meshes on the gpu. *Comput. Graph. Forum 34*, 1 (Feb. 2015), 26–35. 2

[SKU08] SZIRMAY-KALOS L., UMENHOFFER T.: Displacement mapping on the GPU - state of the art. *Computer Graphics Forum 27*, 1 (2008). 2

[SS15] SMITH J., SCHAEFER S.: Bijective parameterization with free boundaries. *ACM Trans. Graph. 34*, 4 (July 2015), 70:1–70:9. 3

[SSCO08] SHAPIRA L., SHAMIR A., COHEN-OR D.: Consistent mesh partitioning and skeletonisation using the shape diameter function. *Vis. Comput. 24* (March 2008), 249–259. 3

[SSGD03] SUNDAR H., SILVER D., GAGVANI N., DICKINSON S.: Skeleton based shape matching and retrieval. In *Proceedings of the Shape Modeling International 2003* (Washington, DC, USA, 2003), SMI '03, IEEE Computer Society, pp. 130–. 2

[SSGH01] SANDER P. V., SNYDER J., GORTLER S. J., HOPPE H.: Texture mapping progressive meshes. In *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques* (New York, NY, USA, 2001), SIGGRAPH '01, ACM, pp. 409–416. 3

[Tar16] TARINI M.: Volume-encoded UV-maps. *ACM Trans. Graph. 35*, 4 (July 2016), 107:1–107:13. 2

[TBTB12] THIERY J.-M., BUCHHOLZ B., TIERNY J., BOUBEKEUR T.: Analytic curve skeletons for 3D surface modeling and processing. *Comput. Graph. Forum 31*, 7pt2 (Sept. 2012), 2223–2232. 3

[TGBE16] THIERY J.-M., GUY E., BOUBEKEUR T., EISEMANN E.: Animated mesh approximation with sphere-meshes. *ACM Trans. Graph. 35*, 3 (May 2016), 30:1–30:13. 3

[THCM04] TARINI M., HORMANN K., CIGNONI P., MONTANI C.: PolyCube-Maps. In *In Proceedings of SIGGRAPH 2004* (2004), pp. 853–860. 3

[TL17] TATARCHUK N., LEFOHN A.: Open problems in real-time rendering. In *ACM SIGGRAPH 2017 Courses* (New York, NY, USA, 2017), SIGGRAPH '17, ACM. 2

[ULP*15] USAI F., LIVESU M., PUPPO E., TARINI M., SCATENI R.: Extraction of the quad layout of a triangle mesh guided by its curve skeleton. *ACM Trans. Graph. 35*, 1 (Dec. 2015), 6:1–6:13. 3, 4

[WYHC14] WONG S.-K., YANG J.-A., HO T.-C., CHUANG J.-H.: A skeleton-based approach for consistent segmentation transfer. *J. Inf. Sci. Eng. 30* (2014), 1053–1070. 3

[XGH*11] XIA J., GARCIA I., HE Y., XIN S.-Q., PATOW G.: Editable polycube map for gpu-based subdivision surfaces. In *Symposium on Interactive 3D Graphics and Games* (New York, NY, USA, 2011), I3D '11, ACM, pp. 151–158. 2

[YBS04] YOSHIZAWA S., BELYAEV A., SEIDEL H.-P.: A fast and simple stretch-minimizing mesh parameterization. In *Shape Modeling Applications, 2004. Proceedings* (2004), IEEE, pp. 200–208. 3, 4

[YHZ*14] YIN K., HUANG H., ZHANG H., GO6NG M., COHEN-OR D., CHEN B.: Morfit: Interactive surface reconstruction from incomplete point clouds with curve-driven topology and geometry control. *ACM Trans. Graph. 33*, 6 (Nov. 2014), 202:1–202:12. 3

[YLPM05] YOON S.-E., LINDSTROM P., PASCUCCI V., MANOCHA D.: Cache-oblivious mesh layouts. *ACM Trans. Graph. 24*, 3 (July 2005), 886–893. 2

[YTSG16] YANG C., TIEBE O., SHIRAHAMA K., GRZEGORZEK M.: Object matching with hierarchical skeletons. *Pattern Recogn. 55*, C (July 2016), 183–197. 2

[ZMT05] ZHANG E., MISCHAIKOW K., TURK G.: Feature-based surface parameterization and texture mapping. *ACM Trans. Graph. 24* (2005), 1–27. 2

[ZSS17] ZAYER R., STEINBERGER M., SEIDEL H.-P.: A gpu-adapted structure for unstructured grids. *Comput. Graph. Forum 36*, 2 (May 2017), 495–507. 2

[ZYH*15] ZHOU Y., YIN K., HUANG H., ZHANG H., GONG M., COHEN-OR D.: Generalized cylinder decomposition. *ACM Trans. Graph. 34*, 6 (Oct. 2015), 171:1–171:14. 3

## Appendix A:

A distance $e$ from a bone for a unit displacement vector $\vec{v}$ is restored perpendicularly using a dot product $p = \vec{v} \cdot \vec{s'}$ (where $\vec{s'}$ is $\vec{s}$ rotated around $\vec{o}$ by the angle $2\pi v'$). The restored displacement vector $\vec{v'}$ is obtained using the formula $\vec{v'} = dp^{-1}\vec{v}$. Afterwards, the distance of a reconstructed point using displacement vector $\vec{v'}$ to the bone is compared to the desired distance $e$. If the two differ, a second case is applied, and the distance is restored using formula $\vec{v'} = c\vec{v}$, where $c$ is obtained according to the cosine and sine theorem as

$$c = \sqrt{a^2 + b^2 - 2ab\cos\gamma}.$$

Here, the $a, b$, and $\gamma$ parameters are

$$\begin{aligned} a &= |W - N| \\ b &= e \\ \gamma &= \pi - \alpha - \beta, \end{aligned}$$

where $\alpha, \beta$ are calculated according to the sine theorem as

$$\begin{aligned} \alpha &= \arcsin\left(\frac{a}{\left(\frac{b}{\sin\beta}\right)}\right) \\ \beta &= \angle(\vec{v}, \vec{g}). \end{aligned}$$

Finally, unknowns $\vec{g}, W$ are computed according to the local $u'$ coordinate as

$$\begin{aligned} W &= H + u'(T - H) \\ \vec{g} &= \begin{cases} -\vec{o}, & \text{if } N = H \\ \vec{o}, & \text{otherwise.} \end{cases} \end{aligned}$$