Eurographics Symposium on Geometry Processing 2011
Mario Botsch and Scott Schaefer
(Guest Editors)

*Volume 30 (2011), Number 5*

# All-Hex Mesh Generation via Volumetric PolyCube Deformation

James Gregson[1], Alla Sheffer[1] and Eugene Zhang[2]

[1]University of British Columbia, Canada
[2]Oregon State University, United States

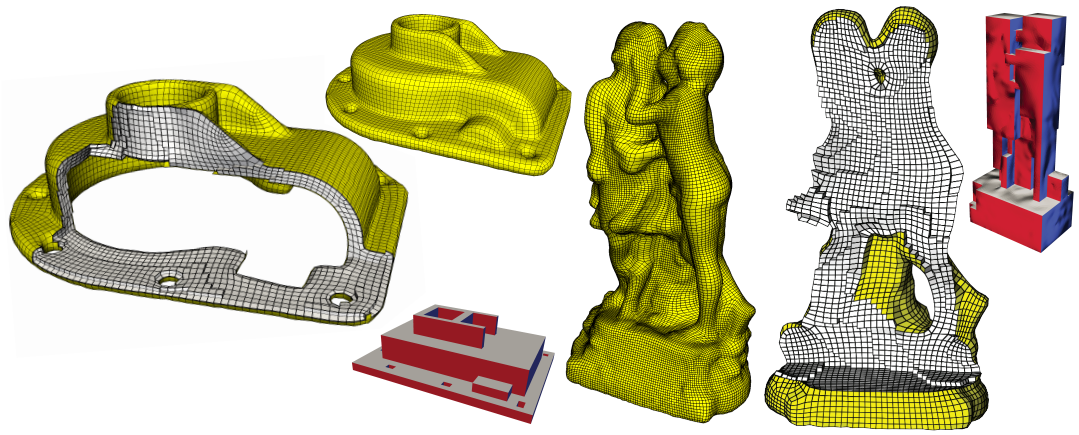


**Figure 1:** *High quality all-hex meshes of complex shapes automatically generated by our method and the PolyCubes we compute to create them. For the* kiss *both fine and coarse meshes are shown.*

**Abstract**

*While hexahedral mesh elements are preferred by a variety of simulation techniques, constructing quality all-hex meshes of general shapes remains a challenge. An attractive hex-meshing approach, often referred to as sub-mapping, uses a low distortion mapping between the input model and a PolyCube (a solid formed from a union of cubes), to transfer a regular hex grid from the PolyCube to the input model. Unfortunately, the construction of suitable PolyCubes and corresponding volumetric maps for arbitrary shapes remains an open problem. Our work introduces a new method for computing low-distortion volumetric PolyCube deformations of general shapes and for subsequent all-hex remeshing. For a given input model, our method simultaneously generates an appropriate PolyCube structure and mapping between the input model and the PolyCube. From these we automatically generate good quality all-hex meshes of complex natural and man-made shapes.*

## 1. Introduction

Due to their numerical properties, hexahedral meshes are preferred and widely used for numerical simulations in several engineering domains [SJ08, Owe09]. However, the automatic meshing of arbitrary objects with high quality, or well shaped, hexahedra remains an open problem [LL10, SJ08]. Consequently, industrial practitioners still largely rely on a variety of semi-manual approaches which require considerable user interaction, and can involve days or even weeks to generate meshes of complex shapes [SJ08].

An appealing approach for all-hex meshing, referred to as mapping or sub-mapping [Owe09], relies on a volumetric mapping between the input model and a PolyCube [THCM04], a solid formed by joining several cubes face-

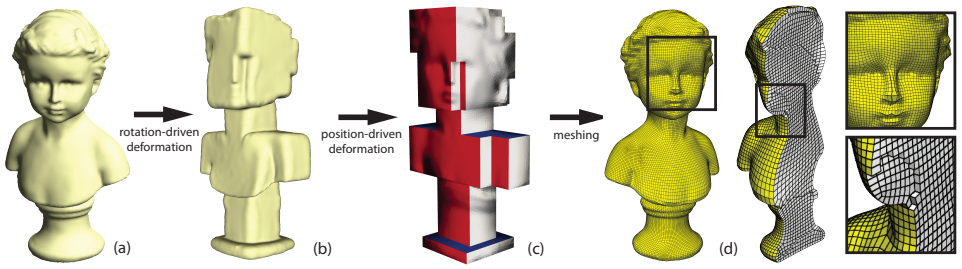



**Figure 2:** *Overview (left to right): Given an input tet mesh (a) we first use a soft, rotation-based, low-distortion deformation framework to align most surface normals with the major axes (b). We then introduce hard positional constraints to obtain the final PolyCube deformation (c). We use the mapping between the input model and the obtained PolyCube to hex-mesh the input with a high quality mesh (d) at a desired fine (left) or coarse (right) resolution.*

to-face which has a trivial hex mesh. Sub-mapping methods transfer the resulting mesh to the input model using the provided mapping. For these methods, hex quality is directly linked to mapping distortion, however computing the PolyCube structure and a low-distortion mapping between the PolyCube and the input remains an open problem for general shapes [XHY*10].

Our work introduces a new method for computing low-distortion PolyCube maps of general shapes and subsequent quality all-hex remeshing based on those (Figures 1, 2). The method uses a volumetric deformation approach to simultaneously compute the Polycube structure and mapping to the input, and is based on the observation that surface normals of a PolyCube are axis-aligned. The method forms the PolyCube by rotating surface normals appropriately, while minimizing the subsequent distortion throughout.

Using the mapping between the input models and the generated PolyCubes (Section 6) we are able to naturally align mesh elements with the input shapes to automatically generate well shaped all-hex meshes of complex natural and man-made shapes (e.g. Figure 1). The resulting meshes have very regular, *structured*, connectivity, and are a good approximation of the input (Section 7).

## 2. Background and Related Work

Our work builds on previous research on mesh generation, PolyCube construction and parameterization, mesh deformation and smooth vector or tensor field computation.

**Mesh Generation:** In recent years production quality methods were developed for generating tetrahedral volume meshes, e.g. [TWAD09, LS07] and surface quad meshes, e.g. [BZK09, DSC09]. However, despite extensive research effort quality hex meshing remains an open problem [SJ08]. Surveys of the existing methodologies are provided by [SJ08, Owe98, Owe09].

While it is fairly easy to generate a basic all-hex mesh (e.g. by subdividing a tet mesh) the main goal of hex-meshing is to generate meshes with reasonable quality in terms of both element shape and connectivity. The element quality, often defined as deviation from a perfect cube [PTS*07], affects the accuracy and robustness of simulations. Simulation results depend not only on average element quality but also on minimum quality [PTS*07, LS07] with even a single "inverted" (negative Jacobian) element making a mesh unusable for simulation. Connectivity also directly impacts computation time; structured meshes with regular connectivity reduce processing time and simplify parallelization significantly [SJ08].

Many hex meshing methods work well for subsets of shapes [MH01, STS05, SOB06] but may not be designed for or robust enough for general inputs [SJ08]. Shepherd [She07] proposes a grid-based method which works well for natural shapes, but requires significant user input for CAD models. Sub-mapping methods [Owe09] used in industry typically rely on an explicit PolyCube structure for the input and so cannot be applied as-is to general inputs. The submapping approach can be extended to general shapes when suitable PolyCubes [LXW*10] are available, however construction of such PolyCubes remains an open problem. In fact, even generating PolyCube domains manually is challenging for users, Han et. al. [HXH10] state "... Thus, it requires the users to be very skillful in designing the parameterization domains", referring to users needing to balance domain simplicity with parameterization distortion.

Consequently, methods used in industry for tend to fall into two categories: hex-dominant and grid-based. Hex-dominant methods, e.g. [VS09, LL10], create high quality meshes but often include a significant percentage of non-hex elements (e.g. often more than 20% for [LL10]). Non-hex elements require specialized numerics and may not suit some applications [Owe98].

Grid or octree methods [Sch96, Mar09] intersect the input model with a Cartesian grid defining the mesh interior. This grid is then connected to the surface using a variety of methods. Grid methods often need excessively fine local element sizes on off-axis or concave features and may form

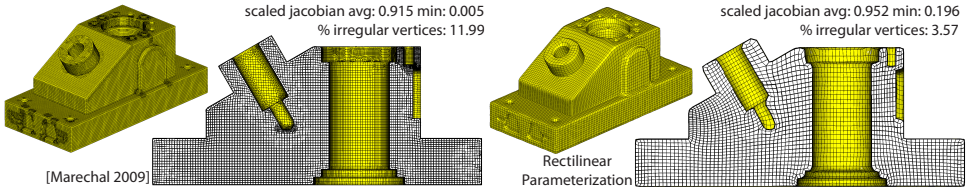



**Figure 3:** *(left) Meshes generated by grid-based techniques, e.g. [Mar09], are poorly aligned with off-axis features, resulting in irregular connectivity and sub-optimal element shape near those ( [Mar09] results kindly provided by the author). (right) Our method naturally aligns the mesh with the boundary surface, improving quality. The biggest improvement shows in the quality of the worst element (a factor of forty in this example) and in the percentage of irregular vertices which drops by a factor of three. [LL10] use this example in their paper as well generating a hex-dominant mesh with over* 20% *non-hex elements and a minimal dihedral angle of* 19°*, our all-hex mesh has a minimal dihedral angle of* 20°.

low quality elements with irregular connectivity in those areas, as illustrated in Figures 3 and 12.

**PolyCube Construction and Parameterization:** PolyCubes in their most general form are volumes bounded by axis-aligned planes. Tarini et al. coined the term PolyCube in [THCM04] and defined a mapping from input objects to the surface of manually constructed PolyCubes. Although recent surface parameterization methods use rectilinear base domains (e.g. [KNP07, BZK09]), the domain structure is generally not applicable to building PolyCubes. So far very few researchers have addressed automatic PolyCube construction. Lin et al. [LJFW08] use a protrusion based segmentation of the input which is approximated by box-primitives, forming very coarse PolyCubes (Figure 4, top-left) with significant distortion. Furthermore their segmentation approach is unlikely to work on CAD models with no such features, e.g. models in Figures 1, 3 and 12. He et al. [HWFQ09] use a distance-based, divide-and-conquer strategy to approximate the input with a PolyCube. Their method is sensitive to off-axis features and may generate over-refined polycubes (see Figure 4) with complex connectivity requiring fine hex meshes for our application. Additionally both methods operate on the surface only, require a separate method to compute the mapping between the volume and PolyCube. This ignores the impact that the PolyCube can have on the distortion of the volumetric mapping in the interior, and computing volumetric mappings from a given PolyCube surface is a challenge in itself, only addressed via complex numerical optimization [LXW*10] or significant user input [XHY*10].

**Deformation:** Existing deformation method for surfaces, surveyed by [Sor06], and volumes, e.g. [BCWG09, BPWG07] try to preserve shape subject to user specified anchor deformation. The user is required to provide feasible deformations for a sparse set of anchors that allow a smooth, meaningful deformation of the input. Our method requires a dense set of anchors over the input surface, so neither space deformation methods [BCWG09] nor multi-resolution approaches [BPWG07] are suitable for our needs. Additionally, since we operate over volumes rather than surfaces, non-linear approaches become prohibitively expensive.

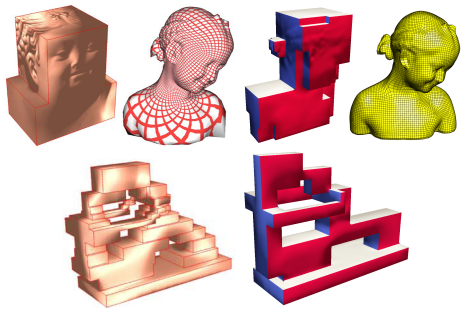


**Figure 4:** *The protrusion-based method of [LJFW08] (top left) produces overly coarse PolyCube maps which introduce significant distortion, while the divide-and-conquer approach of [HWFQ09] (bottom left) oversegments off-axis features and produces unneccessarily complex PolyCube map structure. Our deformation-based approach (right column) performs better and produces PolyCube maps that approximate the inputs well, but are less prone to oversegmentation.*

**Symmetry Tensors:** A natural approach for hex-meshing would seem to be extending state-of-the-art quad meshing algorithms, e.g. [BZK09, DSC09], to volumes. The drawback to these methods for volumetric meshing is the use of symmetry tensors (or cross-fields) that are aligned with principle curvature directions. On surfaces and in 2D, the singular features of these fields are easily handled and determine mesh structure, however singular features in 3D may have very complex features [LL96] that are costly to extract and challenging to resolve in a principled manner.

Our work aims to extend the sub-mapping all-hex meshing approach to general shapes via automatic construction of suitable low-distortion PolyCube maps. By using the automatically computed PolyCube deformation we naturally align the local mesh connectivity with both axis-aligned and off-axis features, creating much more regularly structured,

better quality meshes than the grid-based approaches (Figure 3, right and Section 7).

## 3. Algorithm Overview

The input to our algorithm is an isotropic tetrahedral mesh of the object to be meshed, typically containing 100K to 200K tetrahedra. We generate these meshes with Tetgen [Si] from isotropic surface meshes. Non-isotropic surface meshes are remeshed using Graphite [Lev] prior to tet-meshing.

Our method uses a two step process to deform the input tet mesh into a PolyCube from which an all-hex mesh (Figure 2) is extracted.

**Rotation-Driven Deformation:** The first step of our algorithm gradually aligns the model's surface normals with one of the six global axes ($\pm X, \pm Y, \pm Z$), preserving shape as much as possible (Section 4, Figure 5). This exposes the PolyCube structure and performed in two steps. First, minimal axis-aligning rotations are propagated from the surface through the object volume as deformation gradients. Then, these gradients are integrated using Laplacian deformation to obtain a more axis-aligned deformed object. The first step is carefully designed to prevent singularities from occurring in the deformation gradient field. This property is a key contribution of our method since it allows a volumetric PolyCube structure to be extracted by virtue of unambiguous axis-labeling throughout the object volume.

**Position-Driven Deformation:** After rotation-driven deformation, models are sufficiently axis-aligned to allow a PolyCube structure to be determined (Section 5.1). This structure is extracted and additional positional constraints added to align each PolyCube face with the appropriate axis and enforce planarity (Figure 2, c).

**Mesh Generation:** The final step meshes the resulting PolyCube with an axis-aligned grid. It then maps the mesh back to the input shape using the explicit correspondence provided by the volumetric deformation. The result is a structured all-hex mesh (Section 6, Figure 2, d). Standard mesh improvement techniques are then used to optimize element quality. In the next sections we describe the stages of the pipeline in detail.

## 4. Rotation-Driven Deformation

Our rotation-driven deformation slowly aligns the model surface normals with the global $XYZ$ axes. A gradual, shape preserving process is used to avoid oversegmenting and excessively distorting the final PolyCube (as happens with high-frequency detail when each normal is greedily rotated and enforced strictly). Each iteration of this process produces successively more axis-aligned models (Figure 5) that preserve the overall shape.

### 4.1. Computing Deformation Gradients

Deformation gradients are computed as the minimal rotation necessary to align each surface vertex normal with one of $\pm X, \pm Y, \pm Z$. Gradients are computed for every surface vertex (except those on sharp features) and smoothly propagated to feature and interior vertices to define a volumetric deformation gradient field. Feature vertices are determined by a normal similarity theshold. The use of a two stage deformation scheme is similar to those of [YZX*04, ZHS*05].

We represent deformation gradients with quaternions $\mathbf{q} = [q_x, q_y, q_z, q_w]^T, ||\mathbf{q}|| = 1$ and operate on them using normalized linear interpolation with positive, convex weights. These choices allow standard linear solvers to be used on each component independently when propagating surface gradients smoothly into the object volume. However, the quaternion representation is ambiguous in the sense that $q$ and $-q$ represent the same rotation. This ambiguity admits the possibility of generating zero-norm (or degenerate) quaternions when using linear interpolation. Degenerate quaternions cause highly undesirable singular features within the fields, since they represent the only places where the otherwise smoothly varying orthogonal bases represented by the quaternion field break down.

We suppress this ambiguity by *coherently-orienting* the minimal, axis-aligning rotations used in constructing quaternions. When constructing a quaternion from a rotation matrix $\mathbf{R}$, we first reorder and reflect the columns of $\mathbf{R}$ to maximize its trace. This process guarantees that the scalar components $q_w$ of the quaternions are strictly positive in all cases, which ensures that $||\mathbf{q}|| > 0$ and thus the quaternion is non-degenerate. The use of positive, convex weights then ensures that no quaternion obtained via interpolation (either within a tetrahedron or while propagating gradients) is degenerate, and hence the resulting fields are singularity free. Coherently-orienting the quaternions also keeps rotations close to one another, reducing the approximation error introduced by using linear interpolation rather than more complex, non-linear methods.

The surface deformation gradients are propagated to interior/feature vertices by solving a Laplace equation per quaternion component. The system is solved in a least squares sense with low (unit) weights associated with the surface anchor rotation constraints. Low weights improve robustness to potentially inaccurate initial gradients by gradually enforcing alignment over successive iterations (see Figure 5). Uniform weights are used to discretize the Laplacian operator since the input tet meshes are fairly uniform.

The resulting quaternion field is then normalized to obtain the full set of deformation gradients, and is free of singular features by virtue of the coherent orientation step.

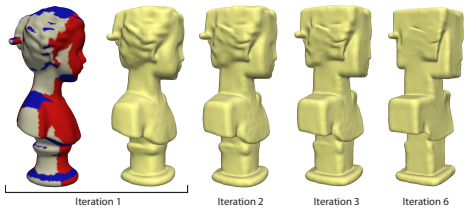



**Figure 5:** *(Left) First deformation iteration: (left) surface gradients, the color shows the target axis for each normal; (right) deformed model. (Right) Consecutive deformation iterations produce progressively more axis-aligned results, converging after roughly four iterations.*

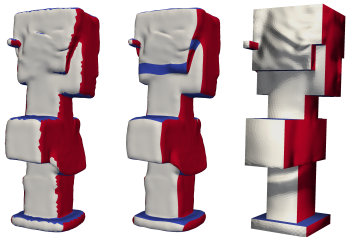


**Figure 6:** *Position-driven deformation, left to right: closest axis for each normal after completing the rotation-driven deformation step, extracted PolyCube structure, final PolyCube deformation.*

### 4.2. Computing Vertex Positions

The deformation gradients are then integrated to obtain a deformed version of the input. For each edge $(i, j)$ with original coordinates $\tilde{v}_i$ and $\tilde{v}_j$ the new vector $v_i - v_j$ can be expressed as $v_i - v_j = \frac{1}{2}(\nabla_i + \nabla_j) \cdot (\tilde{v}_i - \tilde{v}_j)$ where $\nabla_i$ and $\nabla_j$ are the rotation matrices corresponding to the deformation gradients at $i$ and $j$ respectively. These are computed as described in the preceding sections and averaged to approximate the deformation gradient along the edge. Using this formulation for all edges results in a Poisson equation, where the triplet of rows corresponding to the $i$'th vertex is:

$$v_i - \frac{1}{N}\sum_{j=1}^{N} v_j = \frac{1}{N}\sum_{j=1}^{N} \frac{\nabla_i + \nabla_j}{2} \cdot (\tilde{v}_i - \tilde{v}_j). \qquad (1)$$

Here the $j$ index runs over the $N$ neighbors of vertex $i$. Pinning one vertex is sufficient to make equation 1 non-singular. Except for the right-hand-side the system for each coordinate axis is identical, and is solved independently. The resulting deformed model has more pronounced axis-alignment than the input, but is not perfectly axis-aligned since the formulation balances alignment constraint satisfaction with volumetric distortion. The rotation driven deformation step is repeated 5 times, after which it is largely converged (Figure 5). The iterations can be performed efficiently by factorizing and storing the relevant systems.

## 5. Position-Driven Deformation

To complete the PolyCube deformation we extract the PolyCube structure from the deformed meshes and enforce strict planarity constraints on each of the axis-aligned faces, or charts, of the PolyCube. This structure is obvious in most cases, with the deformed models consisting of well-defined largely axis-aligned charts that are separated by nearly-straight boundary edges (Figure 6 left ).

### 5.1. Extracting a PolyCube Structure

**Extracting basic structure:** To extract PolyCube structure we label surface triangles according to the closest axis to smoothed triangle normals and group similarly labeled triangles into charts. Charts correspond to PolyCube faces and the boundaries between charts correspond to PolyCube edges and vertices. We 'straighten' chart boundaries by relabeling any triangle along a chart edge with two neighbors of a common label to the label of the neighbors, which reduces jaggedness. We then remove small, spurious charts bounded by at most two edges (e.g. tiny blue charts in Figure 6, left). Triangles of the deleted charts have their labels reassigned by flood-filling labels from their neighbors. For many models this simple process generates a valid PolyCube structure that can be used as-is for the final position-driven deformation step, however it is often necessary to modify the segmentation by rotating parts of the surface by a fairly large angle to "add" or "delete" charts.

The goal of this segmentation-modifying step is to obtain a segmentation that admits a PolyCube deformation, but unfortunately the required properties of orthogonal polyhedra are not fully known [EM10]. Instead we use a simple greedy heuristic based on the observation that PolyCube edges must be axis aligned and straight. The output from the rotation-driven deformation step is usually close to satisfying this requirement consisting of nearly planar charts with nearly straight edges, however we note two situations where this is not the case. The first is when a single chart should map simultaneously to opposite sides of the final PolyCube, as in Figure 7, which we call a *multi-orientation chart*. This causes the chart to have two U-shaped edges which violate our edge-straightness heuristic. To handle this case, a new chart should be introduced to split the multi-orientation chart into two separated charts, each a single orientation.

The second configuration occurs when charts are *highly non-planar* (such as the red and white charts in Figure 8, left), which introduce S-shaped edges. Such configurations may or may not allow topologically valid PolyCubes to be formed but introduce considerable distortion when planarity constraints are enforced. Highly non-planar charts are also handled by splitting, which reduces distortion once planarity constraints are enforced.

**Splitting multi-orientation charts:** Multi-orientation charts are detected by introducing orientation labels for

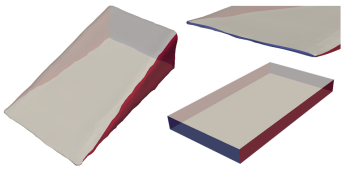


**Figure 7:** *Handling multi-orientation charts: The white chart maps to top and bottom of the PolyCube (left), so a separating chart is introduced (top right) which allows a valid PolyCube to be formed (bottom right)*

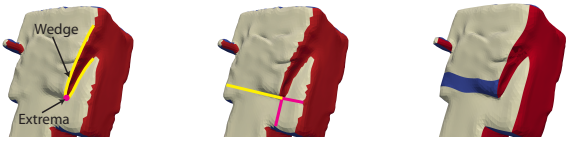



**Figure 8:** *Processing highly non-planar charts (two wedges highlighted): three possible axis-aligned cut options for the red wedge with selected one highlighted (center); produced segmentation (right). In this example only the yellow cut is valid, since the magenta cuts would introduce charts with only two neighbors. Enforcing planarity on the two resulting white patches will no longer collapse the red wedge.*

each triangle to indicate whether its normal points in the positive or negative direction of the axis to which it maps. Charts with multiple orientations are topologically invalid and are split along the boundary where the orientation labels change. A separating chart is introduced along the splitting boundary to separate the two resulting charts (see Figure 7) with width set equal to the user-specified mesh edge-length. The axis-assignment of the separating chart is found by considering its four neighbors. If there are two pairs with the same assignment, the chart is assigned the remaining axis. Otherwise, the assignment from one of the unmatched neighbors is used, with the choice determined by normal similarity. Once detected, multi-orientation charts are split immediately.

**Splitting highly non-planar charts:** The algorithm detects highly non-planar charts by examining the chart edges. It traverses each chart edge looking for "extrema"; points where the edge doubles back on itself. This search is performed on smoothed copies of the chart edges (which are discarded), to avoid detecting spurious extrema introduced by edge jaggedness. Such extrema occur at the end of what we refer to as "wedges" (Figure 8, left), which vary in size depending on how non-planar the adjacent charts are. The severity of an extremum is characterized by the area of its corresponding wedge, which is approximated by the triangle formed by the the extremum itself and its immediately adjacent extremum on either side (or the chart-edge endpoint if there isn't one). Directly enforcing planarity constraints on charts adjacent to wedges flattens the wedges to lines and causes high amounts of distortion. Wedges are resolved by splitting the concave chart starting at the wedge tip and introducing a separating chart along the split.

Extrema are processed in decreasing order of severity. The algorithm finds candidate paths along which to cut the concave chart chart adjacent to the extrema by breadth-first search from each extrema through the chart for boundary vertices of the chart at which to terminate the cut. Valid cuts are defined as those that would not introduce new charts with three or fewer neighbors. Each cut from $a$ to $b$ is assigned a cost $C(a,b) = \alpha L(a,b)(1.1 - Align(a,b))$ based on $L(a,b)$ which is the cut path-length and $Align(a,b)$ which is the maximum dot product of $\frac{b-a}{||b-a||}$ with the global $XYZ$ axes to favor axis aligned cuts. The factor $\alpha$ is 0.5 for cuts that connect two extrema, and 1 otherwise which favors cuts that resolve multiple extrema simultaneously. Once found, a separating chart is introduced along the lowest cost cut. . As before, this separating chart will have four neighbors. If these neighbors have only two axis-assignments among them, the remaining third assignment is used. Otherwise the assignment from the chart forming the wedge is used. The width of the separating chart is set to half of the width of the wedge, defined as 2D distance between the wedge endpoints in the plane corresponding to the two adjacent chart axist assignments (e.g. if the extremum is on an edge between charts assigned to $x$ and $y$, the distance between the endpoints in the $xy$ plane is used.

For all inputs tested, these two splitting rules have produced valid PolyCube structures which allow the subsequent position-driven deformation step to proceed.

### 5.2. Position-Driven Deformation

With the PolyCube structure extracted, the final PolyCube is obtained by constraining each chart to an axis-aligned plane. For this step, an additional variable per-chart (the chart coordinate) is introduced to the Poisson formulation (Equation 1) which all vertices on the chart are constrained to. To avoid self-intersections, which can occur for close-by surfaces aligned with the same axis (see e.g. *casting* model (Figure 1), soft distance preservation constraints are added to preserve the ordering and distance between nearby charts aligned to the same axis. Without these constraints, the ordering of charts can occasionally flip. To measure distance, a per-vertex approximate Voronoi diagram of charts is built within the object volume that stores at the vertices the closest vertex on the closest chart. Edges connecting Voronoi regions of two similarly labeled charts indicate ordering constraints between the charts, with distance equal to the distance along the chart axis between the closest points. These distances are averaged over all edges having a vertex in each Voronoi region of a pair of charts, and used as the constraint distance for the pair. The weight of each constraint is set by a Gaussian function of the distance between the charts $e^{\frac{\max(d_i - \min_d, 0)^2}{\sigma^2}}$, where $\min_d$ is set to twice the output mesh edge-length and $\sigma$ is set to $3\min_d$ to produce a sharp falloff.

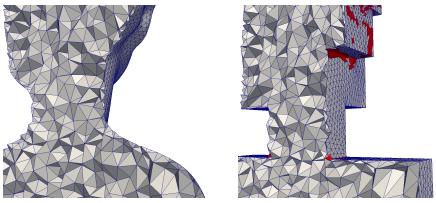


**Figure 9:** *Closeup of the neck of the* girl. *Input mesh (left), mesh after position driven deformation (right). Inverted tets (red) occur when an input tet has four vertices on the surface and at concave edges of the polycube (right). Isolated inversions (not shown) can occur occaisionally in the mesh volume if low-quality sliver tetrahedra from the input invert during deformation.*

Once solved, a set of chart coordinates are obtained. These are rounded to the nearest integer and a Laplace equation solved for each axis, using the known chart coordinates as pin constraints for vertices on each chart. Mean value coordinates in 2D (surface vertices) and 3D (volume vertices) [FKR05] are used to discretize the Laplace operator. The result of this is a PolyCube with the same connectivity as the input, so the mapping between input and output is trivial. This PolyCube is generally not bijective and has overlaps at concave features, however our mesh generation step is designed to handle such cases. Other applications with more stringent bijectivity restrictions could potentially use better input meshes combined with untangling or on-the-fly remeshing such as [WRK*10].

## 6. Mesh Generation

To extract a hex-mesh from the resulting PolyCube, hex vertices and centroids are generated in the PolyCube domain and mapped to the original model using barycentric coordinates. In the PolyCube domain hex corners occur at points with integer coordinates and centers are offset from these by $[\frac{1}{2}, \frac{1}{2}, \frac{1}{2}]^T$. Hex vertices are generated first for the PolyCube corners, then edges, then facets and finally volume. As vertices are generated, those that conflict with the PolyCube structure in their local neighborhood (e.g. volume vertices on the wrong side of a PolyCube facet) are suppressed and discarded, as are vertices for which another vertex already exists with the same parameterization coordinates within five desired edge-lengths. Processing vertices in this order ensures that the necessary information for consistency checks is present when needed. For example, when adding volume vertices, all facet vertices will already be in the lattice, so conflicting facets can be found by examining the lattice positions adjacent to the current vertex. We have found that processing vertices in this manner makes the method robust to a lack of bijectivity in the PolyCube Parameterization. Hex centroids are generated after the vertices in the same way as hex vertices in the PolyCube volume. Output hexahedra are then formed by breadth-first search from each hex centroid for the 8 hex vertices needed to form a hex. The search is performed within the volume of the model. This avoids connecting nearby but disconnected features of the model since hex-centroids will not be generated in these regions, without needing complex intersection tests.

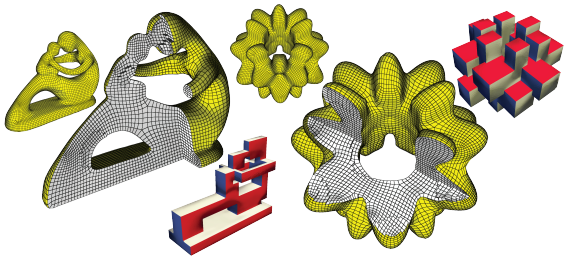
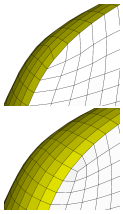


**Figure 10:** *PolyCubes of complex shapes and corresponding hex meshes both generated automatically by our method.*



The resulting meshes are then improved with two standard post processing steps: insertion of a padding layer [She07, Mar09] (inset figure) and mesh optimization. The padding layer is formed by extruding surface quads to form hexes, which provides extra degrees of freedom when hexahedra from PolyCube edges or vertices are placed on smooth parts of the model. Mesh optimization shifts mesh vertices to improve quality, while leaving connectivity unchanged. We reposition vertices individually to maximize the minimum quality of adjacent elements, followed by shape-improvement with the Mesquite software [BDK*03].

## 7. Results and Discussion

We tested our method on a variety of natural shapes and CAD models. Runtimes were generally less than 10 minutes running single-threaded on a Macbook laptop. The challenge for the method is to capture major features while simultaneously determining the PolyCube structure. Figure 1 shows results for the *casting* and *kiss* models, illustrating the method's applicability to inputs with thin features and cases where the PolyCube structure may not be obvious. The girl (Figures 2, 5 and6) is a similarly detailed complex natural model, showcasing our method's ability to capture prominent yet narrow features such as the ponytail (Figure 6). As seen in Figure 10, PolyCubes computed by our method naturally capture the major features of complex models, including off-axis and sharp features of CAD models and intricate details of smooth shapes such as the *bumpy torus*, resulting in regular meshes that capture the input geometry well.

The *hand* (Figure 11) and the camel (Figure 11 ) show our method's behavior on models with multiple elongated features, typical of humanoid and animal shapes. The bunny (Figure 11) is an interesting test model, as its ears are completely misaligned with respect to the global coordinate system, in the standard front-facing orientation we used. While this presents some challenges to our method leading to some

over-segmentation of the PolyCube (Figure 4), the overall mesh quality is still high. The dragon (Figure 11) is the most ambitious model we remeshed. In addition to the casting (Figure 1) we tested the method on a number of other CAD models of varying complexity, containing both smooth and sharp features (Figures 3, 11 and 12), generating high quality meshes which approximate well the input geometries. As shown in Figure 4, PolyCubes generated by our method are better suited for hex-meshing than previous approaches. Our method captures input model features better than [LJFW08] and avoids oversegmenting off-axis features as occurs with [HWFQ09]. This results in meshes that better approximate the input but with very regular connectivity.

With extensive research in hex-meshing, we limit comparison to the most recent fully-automatic methods applicable to general shapes, providing comparisons to two representative hex-dominant [LL10] and grid-based [Mar09] methods.

All of our meshes have positive Jacobian, a minimal requirement for simulation, although this is not guaranteed by the method. To evaluate our output meshes, we use the *scaled Jacobian* metric [PTS*07] which has a range $[-1,1]$ with one being optimal. Our meshes have high minimum and average values for this metric in addition to low percentages of irregularly connected vertices (see Tables 1 and 2). We have also measured the mean and max symmetric Hausdorff distances with [CRS96], and obtained average values of 0.1 and 1.1 edge-lengths (most meshes use an edge length of 1% of the bounding box diagonal), which is comparable to surface remeshing methods. The average of the maximum Hausdorf distances is skewed somewhat by the camel and dragon models which have narrow but long features that are not resolved well by the fixed-size mesh.

Levy and Liu's [LL10] hex-dominant method generates good quality meshes, but with a large percentage of non-hexes. For example, their method produces meshes with 20% and 35% non-hexes for the ANC and dragon models respectively. Our method produces only hex-elements, making it more suitable for many simulators [SJ08].

Figures 3 and 12 compare meshes generated by our method and by the automatic all-hex method of Marechal [Mar09] (who generously provided output meshes, the bounding surfaces of which we used as inputs for our method). Statistics are summarized in Table 2. Our meshes have a lower percentage of irregular vertices (4% vs. $12-29\%$) in addition to higher average quality and significantly higher minimum quality using both scaled Jacobian and Marechal's metrics in all models but the pipe, where our average quality is slightly lower using Marechal's metric (0.830 vs. 0.874). Our method often produces minimum scaled Jacobian values nearly an order of magnitude higher than Marechal's method, while maintaining equal or better average quality. Although the method may round-off sharp edges, the Hausdorff distances from Marechal's meshes are comparable to our other results.

| Model | Num. Hexes | Scaled Jacobian avg. | Scaled Jacobian avg. | avg dist. ($\times 10^{-3}$) | Hausdorff dist. ($\times 10^{-3}$) |
|---|---|---|---|---|---|
| casting | 21k | 0.845 | 0.195 | 1.21 | 5.67 |
| kiss | 215k | 0.920 | 0.125 | 0.325 | 10.2 |
| girl | 194k | 0.925 | 0.235 | 0.481 | 13.2 |
| camel | 16k | 0.901 | 0.169 | 1.18 | 31.5 |
| fandisk | 45k | 0.959 | 0.251 | 1.23 | 6.96 |
| bumpy torus | 35k | 0.891 | 0.270 | 1.21 | 7.86 |
| fertility | 20k | 0.911 | 0.196 | 1.06 | 4.01 |
| bunny | 81k | 0.930 | 0.138 | 1.19 | 4.01 |
| carter | 21k | 0.823 | 0.177 | 1.19 | 6.58 |
| femur | 13k | 0.930 | 0.334 | 1.21 | 9.14 |
| hand | 12k | 0.928 | 0.270 | 0.833 | 11.5 |
| dragon | 173k | 0.882 | 0.002 | 0.694 | 28.9 |
| rocker arm | 18k | 0.899 | 0.226 | 1.16 | 3.95 |
| bimba | 44k | 0.905 | 0.257 | 1.23 | 12.8 |

**Table 1:** *Quality statistics for the models shown in the paper. As demonstrated by these numbers our automatic remeshing algorithm closely approximates both smooth and sharp featured models with good quality all-hex meshes. Meshes were originally generated with an edge length of* $10^{-2}$ *bounding-box diagonals, although the kiss, girl and dragon were subsequently subdivided.*

## 8. Conclusions

We presented a robust method for all-hex mesh generation, capable of producing better quality meshes than previous techniques. Our method significantly improves the quality of the worst mesh element generating meshes that can be largely used as-is by commercial solvers while producing structured meshes which are more efficient for computation.

The key component of our method is the rotation-driven PolyCube deformation step which gradually aligns the normals on the boundary surface of the model with the global coordinate axes while smoothly deforming the mesh interior. Each iteration of the method uses a quaternion deformation gradient propagation approach which generates a singularity-free deformation gradient field in the interior of the domain, enabling all subsequent processing.

**Limitations:** The primary limitations of the method is that, like many automatic all-hex meshing algorithms [Mar09, She07], there are no theoretical guarantees on output mesh quality. Furthermore, the chart insertion process is not guaranteed to produce a valid PolyCube segmentation, it is simply a local greedy heuristic for the global PolyCube consistency problem that works well in conjunction with our Rotation-Driven deformation. However in practice our method is capable of producing high quality output meshes for a wide range of inputs, as illustrated by our results.

The method also has a few limitations which can be addressed by future work. Currently a single-resolution structured mesh is generated, so fine local features can be lost unless sufficiently fine sizing is used (Figure 12, folded-sheet). Using an adaptive meshing scheme similar to [Mar09] could improve this. Second, the results of our method depend on the orientation of the model and would benefit from an automatic method of choosing a suitable orientation. Finally,

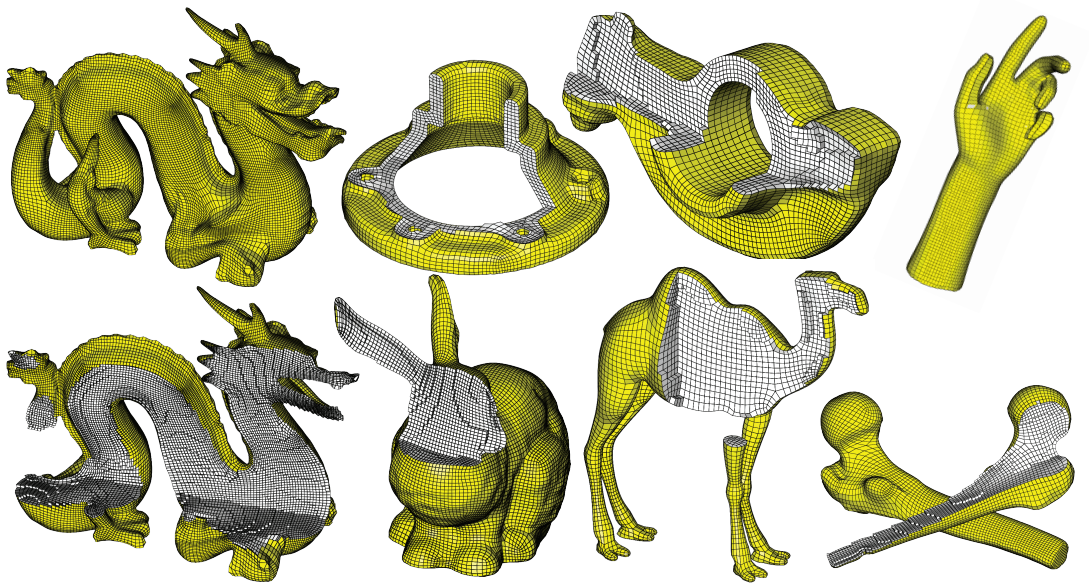


**Figure 11:** *Meshes of a variety of natural and CAD models generated by our method.*

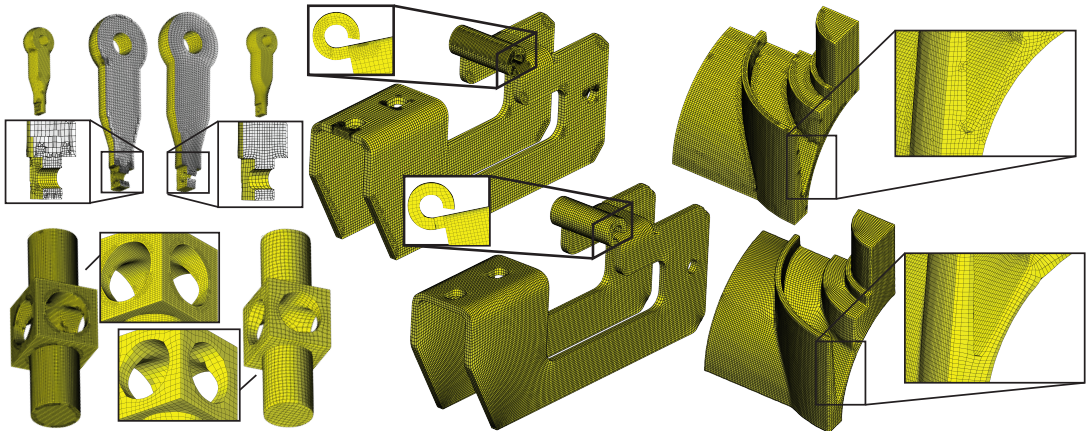


**Figure 12:** *Comparison with [Mar09] (left to right): rod and block, bent sheet, and pipe. Our results are on the right or bottom. Quality statistics listed below. Please zoom into the image using digital version to better see the meshes. On all of these models our method produces more regular meshes leading to better element quality.*

| | Marechal | | | | | | PolyCube deformation | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Model | num. hex | Jac. avg. | Jac. min. | Mar. avg. | Mar. min. | % irr. | num hex | Jac. avg. | Jac. min. | Mar. avg. | Mar. min. | % irr. | Hausdorff dist. (x$10^{-3}$) |
| ANC | 316k | .915 | .005 | .917 | .08 | 12 | 74k | .952 | .196 | .917 | .242 | 4 | 8.59 |
| rod | 18k | .822 | .016 | .850 | .158 | 29 | 18k | .950 | .131 | .928 | .493 | 3 | 5.53 |
| block | 85k | .821 | .018 | .844 | .321 | 27 | 34k | .936 | .215 | .913 | .500 | 3 | 3.23 |
| bent sheet | 62k | .868 | .056 | .833 | .251 | 17 | 99k | .957 | .105 | .904 | .389 | 3 | 10.4 |
| pipe | 52k | .865 | .017 | .874 | .185 | 20 | 84k | .873 | .178 | .830 | .205 | 2 | 5.22 |

**Table 2:** *Quality comparisons with [Mar09]. For each method we measure (left to right): number of elements, Scaled Jacobian: average and minimum, quality metric of [Mar09]: average and minimum, percent of irregular vertices. Haussdorf distances for our meshes with respect to output meshes from [Mar09] are provided in the final column; our meshes were generated using the bounding surface of his meshes as inputs.*

sharp edges may be rounded by our method due to a lack of dedicated sharp feature handling, as seen in Figures 1 (*left*), 3 and 12, resulting in reduced approximation quality.

**Acknowledgements:** We would like to thank the anonymouse reviewers for their helpful comments. Alla Sheffer and James Gregson are funded by NSERC, MITACS and DISCOVERY. Eugene Zhang is partially supported by NSF awards IIS-0546881 and CCF-0830808.

## References

[BCWG09] BEN-CHEN M., WEBER O., GOTSMAN C.: Variational harmonic maps for space deformation. *ACM Trans. Graph. 28* (2009), 34:1–34:11. 3

[BDK*03] BREWER M., DIACHIN L. F., KNUPP P. M., LEURENT T., MELANDER D.: The mesquite mesh quality improvement toolkit. In *Proc. Intl. Meshing Roundtable* (2003). 7

[BPWG07] BOTSCH M., PAULY M., WICKE M., GROSS M. H.: Adaptive space deformations based on rigid cells. *Computer Graphics Forum 26*, 3 (2007), 339–347. 3

[BZK09] BOMMES D., ZIMMER H., KOBBELT L.: Mixed-integer quadrangulation. *ACM Trans. Graph. 28*, 3 (2009), 1–10. 2, 3

[CRS96] CIGNONI P., ROCCHINI C., SCOPIGNO R.: *Metro: measuring error on simplified surfaces*. Tech. rep., Paris, 1996. 8

[DSC09] DANIELS J., SILVA C. T., COHEN E.: Semi-regular quadrilateral-only remeshing from simplified base domains. In *Proc. Symp. on Geom. Proc.* (2009), pp. 1427–1435. 2, 3

[EM10] EPPSTEIN D., MUMFORD E.: Steinitz theorems for orthogonal polyhedra. In *Proc. Symposium on Computational geometry* (2010), SoCG '10, pp. 429–438. 5

[FKR05] FLOATER M. S., KÓS G., REIMERS M.: Mean value coordinates in 3d. *Comput. Aided Geom. Des. 22* (October 2005), 623–631. 7

[HWFQ09] HE Y., WANG H., FU C.-W., QIN H.: A divide-and-conquer approach for automatic polycube map construction. *Computers and Graphics 33* (2009), 369–380. 3, 8

[HXH10] HAN S., XIA J., HE Y.: Hexahedral shell mesh construction via volumetric polycube map. In *Proceedings of the 14th ACM Symposium on Solid and Physical Modeling* (New York, NY, USA, 2010), SPM '10, ACM, pp. 127–136. 2

[KNP07] KÄLBERER F., NIESER M., POLTHIER K.: Quadcover - surface parameterization using branched coverings. *Computer Graphics Forum 26*, 3 (2007), 375–384. 3

[Lev] LEVY B.: Graphite. http://alice.loria.fr/. 4

[LJFW08] LIN J., JIN X., FAN Z., WANG C. C. L.: Automatic PolyCube-maps. In *Advances in Geometric Modeling and Processing* (2008), pp. 3–16. 3, 8

[LL96] LAVIN Y., LEVY Y.: The topology of three-dimensional symmetric tensor fields. In *Late Breaking Hot Topics IEEE Visualization '96* (1996), CS Press, pp. 43–46. 3

[LL10] LEVY B., LIU Y.: Lp centroidal voronoi tesselation and its applications. *ACM Trans. Graph.* (2010). 1, 2, 3, 8

[LS07] LABELLE F., SHEWCHUK J. R.: Isosurface stuffing: fast tetrahedral meshes with good dihedral angles. *ACM Trans. Graph. 26* (2007). 2

[LXW*10] LI X., XU H., WAN S., YIN Z., YU W.: Feature-aligned harmonic volumetric mapping using mfs. *Computers & Graphics 34*, 3 (2010), 242 – 251. 2, 3

[Mar09] MARECHAL L.: Advances in octree-based all-hexahedral mesh generation: handling sharp features. In *Proc. International Meshing Roundtable*. 2009. 2, 3, 7, 8, 9

[MH01] MULLER-HANNEMANN M.: Shelling hexahedral complexes for mesh generation. *Journal of Graph Algorithms and Applications 5*, 5 (2001), 59–91. 2

[Owe98] OWEN S.: A survey of unstructured mesh generation technology. In *Proc. Intl. Meshing Roundtable* (1998). 2

[Owe09] OWEN S.: A survey of unstructured mesh generation technology. http://www.andrew.cmu.edu/user/sowen/survey/hexsurv.html. 1, 2

[PTS*07] PEBAY P., THOMPSON D., SHEPHERD J., KNUPP P., LISLE C., MAGNOTTA V., GROSLAND N.: New applications of the verdict library for standardized mesh verification pre, post, and end-to-end processing. In *Proc. International Meshing Roundtable* (2007). 2, 8

[Sch96] SCHNEIDERS R.: A grid-based algorithm for the generation of hexahedral element meshes. *Engineering with Computers 12* (1996), 168–177. 2

[She07] SHEPHERD J.: *Topologic and geometric constraint-based hexahedral mesh generation*. PhD thesis, University of Utah, 2007. 2, 7, 8

[Si] SI H.: TetGen: A Quality Tetrahedral Mesh Generator and Three-Dimensional Delaunay Triangulator. http://tetgen.berlios.de/. 4

[SJ08] SHEPHERD J. F., JOHNSON C. R.: Hexahedral mesh generation constraints. *Engineering with Computers 24*, 3 (2008), 195–213. 1, 2, 8

[SOB06] STATEN M., OWEN S. J., BLACKER T. D.: Unconstrained paving and plastering: Progress update. In *Proc. International Meshing Roundtable* (2006), pp. 469–486. 2

[Sor06] SORKINE O.: Differential representations for mesh processing. *Computer Graphics Forum 25*, 4 (2006), 789–807. 3

[STS05] SUZUKI T., TAKAHASHI S., SHEPHERD J.: An interior surface generation method for all-hexahedral meshing. In *Proc. International Meshing Roundtable* (2005), pp. 377–397. 2

[THCM04] TARINI M., HORMANN K., CIGNONI P., MONTANI C.: PolyCube-maps. In *Proc. SIGGRAPH* (2004), pp. 853–860. 1, 3

[TWAD09] TOURNOIS J., WORMSER C., ALLIEZ P., DESBRUN M.: Interleaving delaunay refinement and optimization for practical isotropic tetrahedron mesh generation. *ACM Trans. Graph. 28* (2009). 2

[VS09] VYAS V., SHIMADA K.: Tensor-guided hex-dominant mesh generation with targeted all-hex regions. In *Proc. International Meshing Roundtable* (2009). 2

[WRK*10] WICKE M., RITCHIE D., KLINGNER B. M., BURKE S., SHEWCHUK J. R., O'BRIEN J. F.: Dynamic local remeshing for elastoplastic simulation. *ACM Trans. Graph. 29* (July 2010), 49:1–49:11. 7

[XHY*10] XIA J., HE Y., YIN X., HAN S., GU X.: Direct-product volumetric parameterization of handlebodies via harmonic fields. In *Proc. Shape Modeling Intl.* (2010), pp. 3–12. 2, 3

[YZX*04] YU Y., ZHOU K., XU D., SHI X., BAO H., GUO B., SHUM H.-Y.: Mesh editing with Poisson-based gradient field manipulation. In *Proc. SIGGRAPH* (2004), pp. 644–651. 4

[ZHS*05] ZHOU K., HUANG J., SNYDER J., LIU X., BAO H., GUO B., SHUM H.-Y.: Large mesh deformation using the volumetric graph Laplacian. *Trans. Graph. 24* (2005), 496–503. 4