Pacific Graphics (2017)
J. Barbic, W.-C. Lin, and O. Sorkine-Hornung (Editors)

Short Paper

Robust Edge-Preserved Surface Mesh Polycube Deformation

Hui Zhao[1] Na Lei[2,3†] Xuan Li[4] Peng Zeng[1] Ke Xu[5] Xianfeng Gu[4]

[1]Tsinghua University, China
[2]DUT-RU ISE, Dalian University of Technology, China
[3]Key Laboratory for Ubiquitous Network and Service Software of Liaoning Province, China
[4]State University of New York at Stony Brook, USA
[5]Beijing University of Technology, China

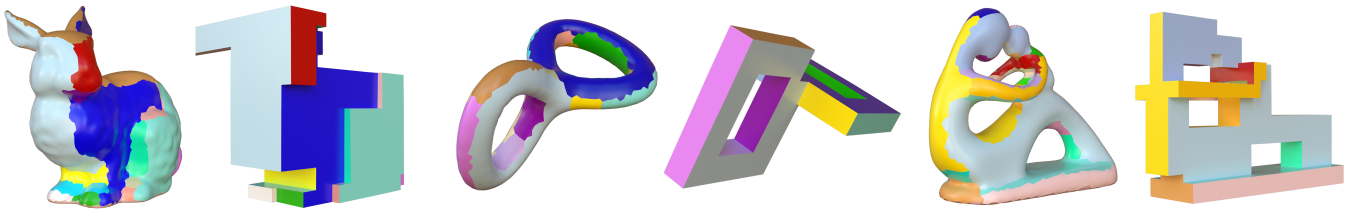

Figure 1: *Three surface models and their corresponding polycube shapes*

Abstract

The problem of polycube construction or deformation is an essential problem in computer graphics. In this paper, we present a robust, simple, efficient and automatic algorithm to deform the meshes of arbitrary shapes into their polycube ones. We derive a clear relationship between a mesh and its corresponding polycube shape. Our algorithm is edge-preserved, and works on surface meshes with or without boundaries. Our algorithm outperforms previous ones in speed, robustness, efficiency. Our method is simple to implement. To demonstrate the robustness and effectiveness of our method, we apply it to hundreds of models of varying complexity and topology. We demonstrate that our method compares favorably to other state-of-the-art polycube deformation methods.

CCS Concepts

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

1. Introduction

Polycube is coined and firstly proposed in [THCM04] to extend cube mapping to general shapes. This kind of special shapes generalize Geometry Images [GGH02] to allow geometry and texture stored efficiently.

In this paper, we propose a novel, automatic polycube deformation algorithm applied on surface mesh. Our framework separates the polycube construction process into three components explicitly: segmentation, polycube topology and polycube geometry. We use existing algorithm for the first and the second steps. Our major contribution is on the polycube geometry step. The technique we implement can process all kinds of meshes: such as different genus, orientation or non-orientation, with or without boundaries. There is no pre-processing or post-processing cleanup processing in our method. As we achieve the polycubes by deforming original meshes, we get a direct cross-surface parameterization between the meshes and their corresponding polycube shapes automatically.

2. Related Works

Recently some automatic algorithms are proposed in [HWFQ09, LJFW08, GSZ11, HJS*14]. The authors in [LJFW08] uses a segmentation method to patch the input mesh, then use box-primitives to approximate it coarsely . But this method fails in complicated models. [HWFQ09] applies a distance-based, divide-and-conquer algorithms to build the polycube. The method in [HWFQ09] generates over-refined polycubes and is sensitive to off-axis features. The algorithms in [HWFQ09,LJFW08] are based on surface meshes and can not build the cross-surface map automatically. While the algorithms in [GSZ11, HJS*14] are volume mesh based, these two

† Corresponding author: nalei@dlut.edu.cn

DOI: 10.2312/pg.20171319

algorithms look for a specific polycube which minimizes the distortion of the volumetric map. In [FBL16], they propose another polycube method which is similar to ours. Their algorithm is also a normal-driven method.

Poisson system based deformation [YZX*04] is well-known technique. After the rotations of all triangle faces are known, the triangles can be rotated into the new orientation, then the Poisson system is used to blend the triangle soup together and reconstruct a consistent mesh into its new shape.

3. Our Methods

Polycube is also called as orthogonal polyhedra [LVS*13, EM10]. We observe that there are three components in deforming a mesh into its polycube shape: segmentation, polycube topology and polycube geometry.

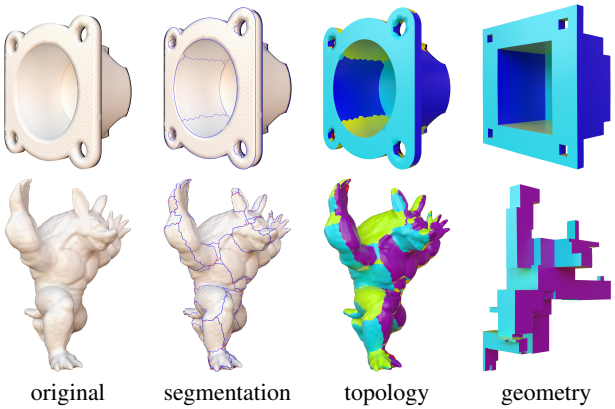

Figure 2: *The segmentation, polycube topology and geometry.*

These three steps can be independent from each other. The first step divides a mesh into several different patches. The second step determines the polycube topology of the mesh, and the third step fixes the polycube geometry. In Figure 2, the two models are segmented into some parts as the second column; then based on the parts, we assign every model a polycube topology in the third column; finally our algorithm can obtain an exact polycube geometry in the last column.The previous algorithms mix these two or three steps into one. Our algorithm separates them explicitly. In this paper, we focus on the polycube geometry step. Given a model with a valid polycube topology, such as using the method proposed in [FBL16], our algorithm produces a final shape with perfect polycube geometry.

In the first step, it is well known the necessary conditions of the segmentation for a valid polycube is still an open problem [EM10]. However there are three sufficient conditions [EM10] :

a) one single patch of a polycube has at least four other neighboring charts;

b) two neighboring polycube patches must not have opposite labels;

c) the valence of every polycube vertex is three.

In this paper, we use the same polycube topology validated data in [FBL16] for the comparison.

3.1. Polycube Topology

The second step is determining the polycube topology. Given a valid segmentation of a mesh, this step labels or associates each triangle with one of six axis $(+X,-X,+Y,-Y,+Z,-Z)$, such as in Figure 2, the six different colors represent $(+X,-X,+Y,-Y,+Z,-Z)$ respectively.

A valid polycube topology assigns a target normal to every triangle face. In each patch, all triangles have the same target normal. Our algorithm rotates all triangles to their corresponding target normal directions. There is no explicit constraints between the patches. Every triangle is independently rotated. However there is an implicit global topological constraints between them due to the polycube topology.

Different polycube topologies. If the same model is assigned several different polycube topologies, then we will have different polycube shapes. In the Figure 3 and 4, we demonstrate this conclusion. The first and the third columns of Figure 3 show the same "bimba" model, but with different polycube topologies, and the second and the last columns are their corresponding polycube shapes.

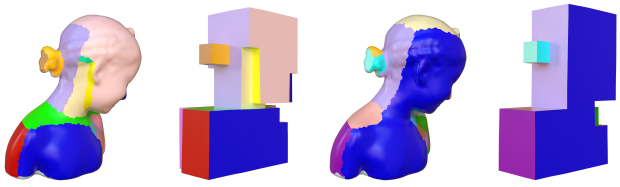

Figure 3: *The model and two different polycube topologies.*

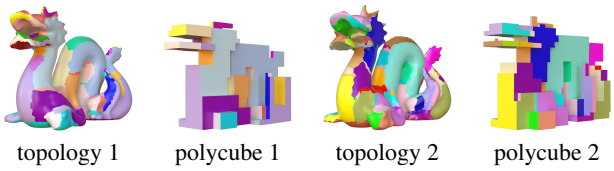

Figure 4: *Another model and two different polycube topologies.*

3.2. Polycube Geometry

This step aligns and reorients the triangles in each patch with one axis direction. And every patch should be mapped into a planar polygon and all of chart boundaries are straight lines.

Our polycube geometry method is based on a Poisson system which reconstructs the deformed polycube mesh to satisfies the current assigned face normals of triangles. As the Poisson system can only approximate the input normal requirements, therefore we use an iterative Poisson systems. After several iteration, our system converges and outputs the corresponding polycube shape whose boundary between two patches is exactly straight automatically,

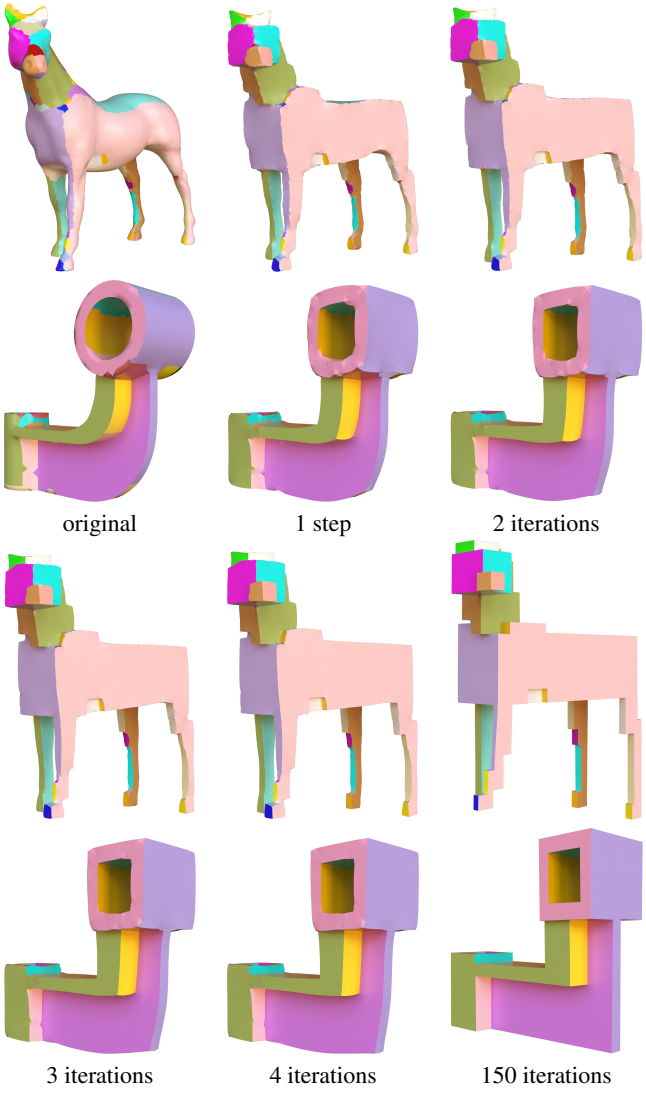

Figure 5: *The iteration of the polycube deformation of two models.*

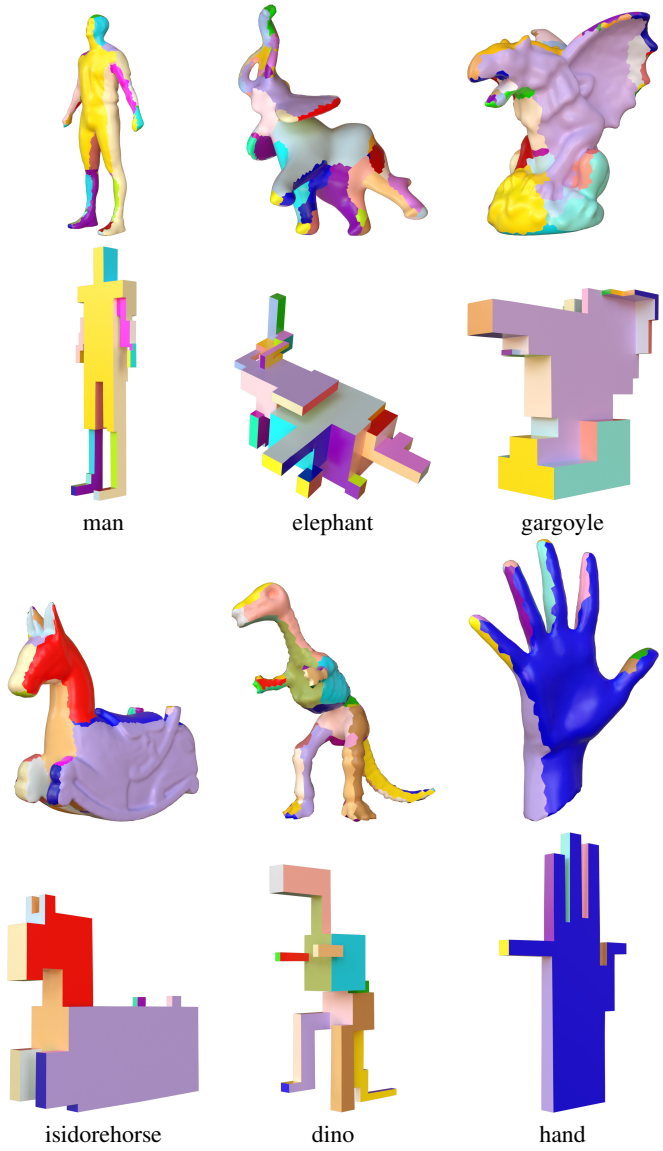

Figure 6: *The six models and their polycube shape.*

and the triangles in every chart fall on a plane automatically without the need of any extra planarity constraints.

Motived by their explanation and derivation on stretching deformation energy [ZG16]. Our algorithm adopts and modifies their stretching energy. In their original method [ZG16], the rotations of faces are unknown variables and changed in every iterative step, however in our configuration, the rotations are known in advance and kept the same in our iteration. In fact, our deformation does preserve the metric in theory, but in practice we observe the change of edge length is small.

Let S be an original surface and S' be its deformed surface embedded in 3-dimension. And denote a 3-vector x_v be the position associated with vertex v of S, and a 3-vector x'_v with vertex v of of S'. On every triangle of the mesh, we define one rotation matrix variable referred to as $R(t)$. The stretching energy [ZG16] is defined as

$$E(x',R) = \sum_{he_{vw}} \cot(a_{vw}) \| (x'_v - x'_w) - R(t_{vw})(x_v - x_w) \|^2. \quad (1)$$

In above, $\|\cdot\|^2$ is the standard 3-vector norm, he_{vw} represents the half edge from the vertex v to w. We denote the angle of the corner opposite to the half edge he_{vw} in its triangle with a_{vw}. Finally $R(t_{vw})$ represents the 3×3 rotation matrix associated with the triangle face whose the half edge is he_{vw}.

It is proved in [ZG16] that the $E(x')$ measures the quantity

$$\int_s (\sigma_1(p) - 1)^2 + (\sigma_2(p) - 1)^2 \; dA_g(p), \quad (2)$$

where the symbol $\sigma_1(p)$ and $\sigma_2(p)$ represent the the maximum and

minimal stretching ratios of a tangent vector of S at a point p under the differential mapping dx' from S to R^3. Therefore $E(x')$ is one reasonable quantity to measure the stretching of a deforming surface.

This stretching energy is quadratic in x' with a fixed rotation matrix R per triangle. Take the gradient of the stretching energy and set it to zero. We can obtain the optimal variables x' by solving a single linear system as the following:

$$\sum_{w\in N(v)} \left[\cot(a_{vw})+\cot(a_{wv})\right](x'_v - x'_w) = \sum_{w\in N(v)} \left[\cot(a_{vw})R(t_{vw})+\cot(a_{wv})R(t_{wv})\right](x_v - x_w). \quad (3)$$

By defining the 3-vector at vertex v as:

$$b_v := \sum_{w\in N(v)} \left[\cot(a_{vw})R(t_{vw})+\cot(a_{wv})R(t_{wv})\right](x_v - x_w), \quad (4)$$

we can change the above system into matrix format as:

$$Lx' = b, \quad (5)$$

where L is the n-by-n Laplacian matrix, x' and b are n-vectors of 3-vectors.

Rotation per triangle. In the above system, we need know the rotation matrix of every triangle face of the mesh. Although we do not know the exact vertex positions of the polycube in advance, the face normals of the polycube are determined and fixed by its polycube topology. Therefore we can calculate the rotation matrix for every triangle from its unit normal on the original mesh and the target normal from its polycube topology without knowing the target polycube shape. Given two unit normals, the rotation between them can be computed by the algorithm of Rodrigues' rotation formula.

The stretching energy defined in equation 1 can measure the stretching ratio if it is a function of both rotation R and unknown position vectors x'. In our framework, we fix the variable R, our system is only a function of unknown position vectors x'. Therefore our method does not minimize the stretching energy, it is a simple Poisson system. The explanation of "stretching energy" in [ZG16] gives us a hint why our simple Poisson system changes edge length not so much in practice.

Iteration. Poisson system basically is an approximation method. The system 5 cannot result in an exact polycube, such as the models with a nearly polycube shape shown in Figure **??**. We fix the problem with an iteration method. In every step i, we recompute the rotation matrix $R_i(t)$ for the triangle t according the face normal of the current model and the target normal from its polycube topology. With the new $R_i(t)$, we update the L_i and b_i. Then the iteration system is as the following:

$$L_i x'_i = b_i\,. \quad (6)$$

Figure 5 demonstrates the iteration process of the two polycube deformation. We observe that the polycube shapes get better and better after each iteration. And the planarity and straightness of the polycube are realized in the converge of the iterations.

In these experiments, It is shown that the polycube shape in the fifth step is almost the same as the one from the hundredth step. Therefore the converge speed of our polycube deformation is very fast. In practice, the speed varies according to the different models.

4. Results and Demonstrations

In Figure 6, we show several models and their polycube shapes. The technique we propose can manipulate models with high genus directly, such as shown in Figure 14.

Figure 7 exhibit some meshes with boundaries and their polycubes. As there are implicit constraints of polycube topology, the edges on the boundary can not be deformed into the straight lines.

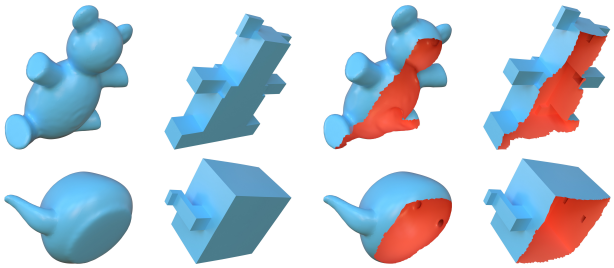

Figure 7: *The models with or without boundary, and their polycube shapes.*

Even on non-orientable meshes, our algorithm can also deform them successfully. In Figure 8, the well-known "costa" surface mesh is deformed into a polycube.

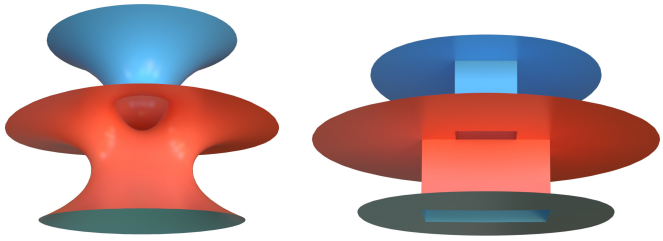

Figure 8: *The non-orientable surface "costa" and its polycube.*

Comparison. In this part, we compare the methods between ours and theirs [FBL16]. We use the same models, the same segmentation charts and the same polycube topologies of the data in [FBL16]. We run our algorithms in a hundred of models, and exhibit several results in Figure 9 and Figure 10.

The polycube shapes from our and their algorithms are almost the same. However the area and the size of every polycube face is slightly different. We compute the edge and area errors for each model with our and their polycube results. The error ratios of one hundred models from two methods are displayed in Figure 11 and 12. We can conclude that our algorithm can preserve the edge and area much better than the method in [FBL16].

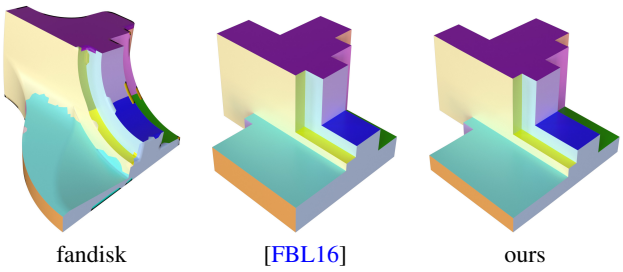

Figure 9: *The polycube comparison of [FBL16] and ours.*

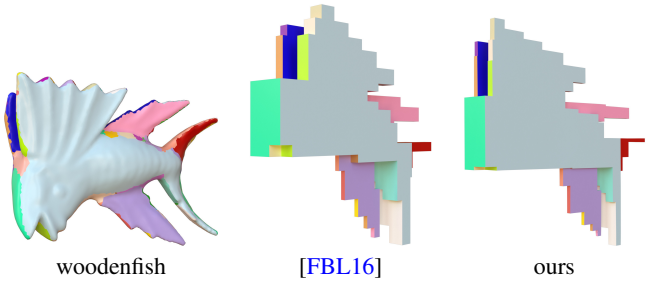

Figure 10: *The polycube comparison of [FBL16] and ours.*

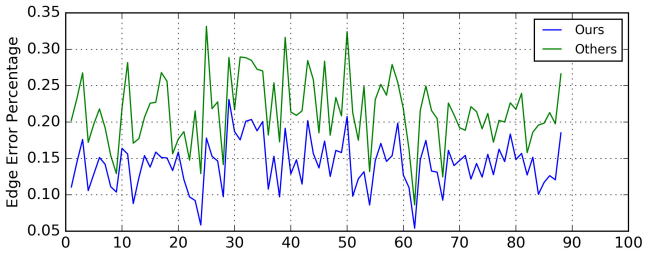

Figure 11: *The edge error diagram of our algorithm and [FBL16].*

s

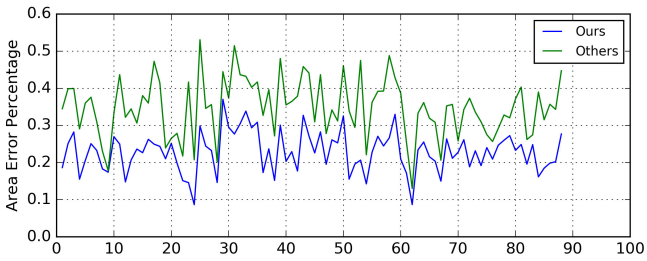

Figure 12: *The area error diagram our algorithm and [FBL16].*

5. Conclusion and Future work

In this paper, we present a robust, efficient polycube deformation algorithm. The quadrangulation and hexahedral meshing from a surface mesh is a crucial problem in graphics community, it is a promising direction to exploit our method in these kinds of applications.

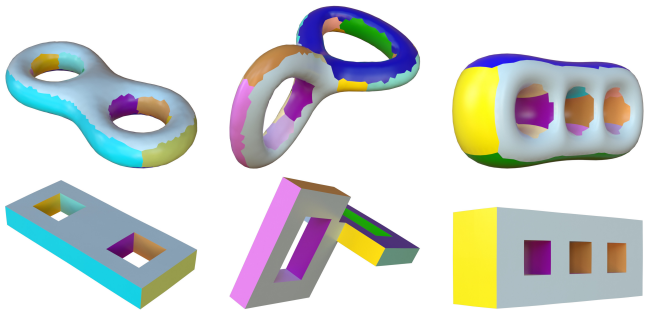

Figure 13: *The models of high genus and their polycube shapes.*

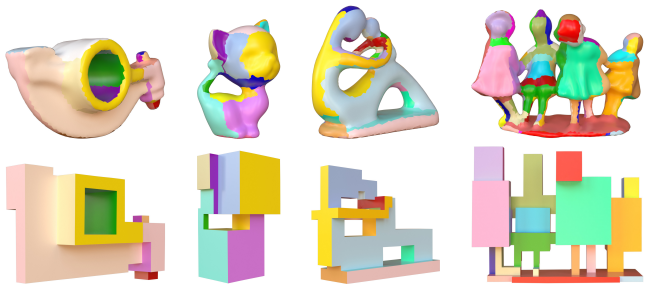

Figure 14: *The models of high genus and their polycube shapes.*

Acknowledgments

We wish to thank anonymous reviewers for encouragements and thoughtful suggestions. We are grateful for Professor Steven J. Gortler for the motivation and the insightful guide which make this paper possible. We also thanks Yue Li for the help in our experiments. Mesh models are courtesy of the Aim@Shape Repository, the Stanford 3D Scanning Repository and the dataset of [FBL16]. We used Mitsuba [Jak10] for rendering images. Our algorithms are implemented on MeshDGP [Zha16] framework. We also thank Libigl [JP*16] for reference. The project is partially supported by NSFC 61772105, 61720106005, 11271156, NSF DMS-1418255 and AFOSR FA9550-14-1-0193.

References

[EM10] EPPSTEIN D., MUMFORD E.: Steinitz theorems for orthogonal polyhedra. In *Proceedings of the twenty-sixth annual symposium on Computational geometry* (2010), ACM, pp. 429–438. 2

[FBL16] FU X., BAI C., LIU Y.: Efficient volumetric polycube-map construction. *Computer Graphics Forum (Pacific Graphics) 35*, 7 (2016). 2, 4, 5

[GGH02] GU X., GORTLER S. J., HOPPE H.: Geometry images. In *ACM Transactions on Graphics (TOG)* (2002), vol. 21, ACM, pp. 355–361. 1

[GSZ11] GREGSON J., SHEFFER A., ZHANG E.: All-hex mesh generation via volumetric polycube deformation. In *Computer graphics forum* (2011), vol. 30, Wiley Online Library, pp. 1407–1416. 1

[HJS*14] HUANG J., JIANG T., SHI Z., TONG Y., BAO H., DESBRUN M.: L1-based construction of polycube maps from complex shapes. *ACM Transactions on Graphics (TOG) 33*, 3 (2014), 25. 1

[HWFQ09] HE Y., WANG H., FU C.-W., QIN H.: A divide-and-

conquer approach for automatic polycube map construction. *Computers & Graphics 33*, 3 (2009), 369–380. 1

[Jak10] JAKOB W.: Mitsuba renderer, 2010. http://www.mitsuba-renderer.org. 5

[JP*16] JACOBSON A., PANOZZO D., ET AL.: libigl: A simple C++ geometry processing library, 2016. http://libigl.github.io/libigl/. 5

[LJFW08] LIN J., JIN X., FAN Z., WANG C. C.: Automatic polycube-maps. In *Advances in Geometric Modeling and Processing*. Springer, 2008, pp. 3–16. 1

[LVS*13] LIVESU M., VINING N., SHEFFER A., GREGSON J., SCATENI R.: Polycut: Monotone graph-cuts for polycube base-complex construction. *Transactions on Graphics (Proc. SIGGRAPH ASIA 2013) 32*, 6 (2013). doi:10.1145/2508363.2508388. 2

[THCM04] TARINI M., HORMANN K., CIGNONI P., MONTANI C.: Polycube-maps. *ACM Transactions on Graphics (TOG) 23*, 3 (2004), 853–860. 1

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

[ZG16] ZHAO H., GORTLER S. J.: A report on shape deformation with a stretching and bending energy. *CoRR abs/1603.06821* (2016). URL: http://arxiv.org/abs/1603.06821. 3, 4

[Zha16] ZHAO H.: MeshDGP: A C Sharp mesh processing framework, 2016. http://meshdgp.github.io/. 5