Robust Edge-Preserved Surface Mesh Polycube Deformation

Hui Zhao¹ Na Lei² Xuan Li² Peng Zeng¹ and Ke Xu³ Xianfeng Gu²

¹Tsinghua University, China
²State University of New York at Stony Brook, USA
³Beijing University of Technology, China
⁴Dalian University of Technology, China

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 denotes a special geometric shape whose face normals are aligned with the one of the six axis of a specific orthonormal coordinate frame. A polycube captures overall and global shape features of a mesh, and remove its local details.

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. Due to its highly regular structure and special global parametric domain, polycube is applied in many kinds of graphics applications, such as: surface texturing [THCM04], volumetric texturing [CL*10], parameterization [YL08;GXH*13], reconstruction [WJH*08], hexahedral remeshing [GSZ11;XHY*10;HXH10;LV*13], shape morphing [FJFS05], spline constructions [WHL*08;WHI*08], volumetric mapping [LGW*07;HYLG08] and T-mesh construction [LZLW15].

Constructing polycube shapes from meshes is a challenging problem. In earlier works, the polycube meshes are constructed manually [THCM04;YL08], which needs a lot of tedious, labor-intensive user interaction with great care. After the polycube is available, these methods need another extra algorithm to achieve the cross-surface map between the polycube and the mesh [WHL*08]. The cross-surface map is also a challenge problem itself [WJH*08]. If we change the polycube, we need rebuild the map.

In this paper, we propose a novel, automatic polycube deformation algorithm applied on surface mesh. Our method separates the polycube construction process into three components explicitly: segmentation, polycube topology and polycube geometry. Our major contribution is on the polycube geometry step, we propose a face normal rotation based deformation method. The technique we implement can process all kinds of meshes: such as different genus, orientation or non-orientation, with or without boundaries. Compared to previous methods, our algorithms is more efficient, robust, fast, accurate and can deform an arbitrary mesh of complicated ge-
ometry and arbitrary topology into its polycube. There is no pre-processing or post-processing cleanup processing, and topological degeneracies in our experimental results. 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, HJS14]. The authors use a segmentation method to patch the input mesh, then use box-primitives to approximate it coarsely in [LJFW08]. 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 [HJS14] are based on surface meshes and can not build the cross-surface map automatically. While the algorithms in [GSZ11] are volume mesh based, these two algorithms look for a specific polycube which minimizes the distortion of the volumetric map.

Polycube is a an axis aligned shape which mimics the original shape, but with a geometric characterization that its every face normal is aligned with one of the axis of an orthonormal coordinate frame respectively. Therefore the value of the $\ell_1$-norm of every unit face normal of the polycube shape is equal to 1 [HJS14].

Based on this observation, [HJS14] defines a $\ell_1$-norm energy which is weighted by triangle areas, then proposes a variational method to deform an input triangle mesh into its polycube shape by the minimization of this energy. The $\ell_1$ norm term of the mesh’s face normals and the weight term of triangle areas both are nonlinear in the mesh position. Then they change the unconstrained system into a constrained minimization to make it well-behaved, yet numerically tractable and efficient. Finally complicated numerical method is resorted to solve the minimization problem in [HJS14]. To decrease the distortion, a volumetric mesh is created from surface mesh. Then an as-rigid-as-possible volumetric distortion energy [ACOL09] is used to regularize the system. There are a lot of minimum for the deformation energy, therefore one regularization approach is also used to single out the desired polycube [HJS14].

Because the resulting polycube is different with varying orientation, they also introduce an energy term to find the optimal global orientation for the polycube. This energy is integrated into their whole system. However, the optimal polycube in all orientation is an ill-defined concept. We do not think there should be an optimal polycube.

The polycubes achieved from above minimization often have some spurious topological degeneracies. A post-processing cleanup step is used to fix the problem in [HJS14]. Their method also requires the input mesh is closed, however our method can process open boundary meshes.

The method in [GSZ11] proposes to align the surface normals of an input mesh with one of six axes($\pm X, \pm Y, \pm Z$) gradually, this step is called rotation-driven deformation. However the result can not be a perfect polycube and second position-driven deformation is required to align each polycube face with the corresponding axis exactly and enforce planarity. This method is similar to ours, but our algorithms uses different method to compute rotations, therefore ours can converge to enforce the faces to be planer and do not need other post-processing steps.

The algorithm in [WYZ11] requires an existing polycube, then they can optimize it to match the desired quality. This method in [LVS13] attacks the polycube segmentation problem. They use graph-cut based approach to achieve a polycube base-complex to satisfy certain quality requirement. The algorithms can balance parameterization distortion against the singularity count of a polycube. We use the same segmentation step as theirs. [LVS13, FBL16], we propose a different polycube geometry deformation method from theirs.

In [FBL16], they propose another polycube method which is similar to ours. Their algorithm is also a normal-driven method. Given a polycube with complicated topology, a simplification method is proposed in [CLS16].

Poisson system based deformation [YZX04] 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. The rotations can be computed according to the different application. In [XZWB06] the rotation is achieved by the interpolation from two corresponding meshes. In [ZKKS05], they interpolate the rotations of triangles.

3. Our Algorithms

Polycube is a shape formed by joining several rectangle face against face and the surface normals of a polycube are axis-aligned. Polycube is also called as orthogonal polyhedra [LVS13, EM10]. We observe that there are three components in deforming a mesh into its polycube shape: segmentation, polycube topology and polycube geometry.

These three steps can be independent from each other. The first step divides a mesh into several different charts. 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 in Figure 2a, then based on the parts, we assign every model a polycube topology in Figure 2b, finally our algorithm can obtain an exact polycube geometry in Figure 2c. 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, the whole mesh is separated into several charts and the chart boundary is monotone [LVS13]. 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.

In the segmentation step, we must guarantee that there are only three parts which meet on a point to satisfy the above third topological requirement. The PolyCut method in [LVS∗13] can be applied in this step. In this paper, we use also the same polycube topology validated data in [FBL16] for the comparison.

3.1. Polycube Topology

The second step is determining the polycube topology. After a specific 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 2c, the six different colors represent (+X,−X,+Y,−Y,+Z,−Z) respectively.

A valid polycube topology assigns a target normal to every triangle face, and divides the whole mesh into different patches. In each patch, the 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 patch 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 Figure 3 and 4 we demonstrate this conclusion. Figure 3a and 3c show the same "bimba" model, but with different polycube topologies, and Figure 4a and 4c are their corresponding polycube shapes.

3.2. Polycube Geometry

The second step aligns and reorients the triangles in each chart with one axis direction. And every chart should be mapped into a plane rectangle and all of chart boundaries are straight lines.

When creating a polycube from a mesh, we wish that the parameterization distortion is low. At the same time, the number of singularity, i.e. the chart corners and the number of charts is kept low. The algorithm of polycube construction should provide an optimal trade-off between parametrization distortion and chart or singularity numbers.

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

Changing a model into its polycube shape fundamentally is a surface deformation process. Previously all kinds of deformation algorithms [ZG16, CPSS10, BS08, SA07] focus on preserving the local features of the original model as much as possible. In polycube case, we propose that the target of the deformation is preserving the metric instead of local features. In [ZG16], the deformation energy is separated into two kinds of energy explicitly: stretching and bending energy. The stretching energy tries to preserve the metric, and the bending energy preserves the mean curvatures. Motivated by their explanation and derivation on stretching energy [ZG16]. Our algorithm adopts and modifies their stretching energy under the constraints of the target normal direction of every triangle of the polycube topology. 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 ad-
vance and kept the same in our iteration. In fact, our deformation does not 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

$$E(x',R) = \sum_{hev \in \mathcal{E}} \cot(a_{vw})\| (x'_v - x'_w) - R(t_{vw})(x_v - x_w) \|^2.$$  \hspace{1cm} (1)

In above, $\| \cdot \|^2$ is the standard 3-vector norm, $hev_{vw}$ 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 $3 \times 3$ 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
\[
\int \left( (\sigma_1(p) - 1)^2 + (\sigma_2(p) - 1)^2 \right) dA_\mathcal{R}(p),
\]
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 $\mathbb{R}^3$. 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:
\[
\sum_{w \in \mathcal{N}(v)} \left[ \cot(a_{vw}) + \cot(a_{vw}) \right] (x'_v - x'_w) = \sum_{w \in \mathcal{N}(v)} \left[ \cot(a_{vw})R(t_{vw}) - \cot(a_{vw})R(t_{vw}) \right] (x_v - x_w).
\]

By defining the 3-vector at vertex $v$ as:
\[
b_v := \sum_{w \in \mathcal{N}(v)} \left[ \cot(a_{vw})R(t_{vw}) + \cot(a_{vw})R(t_{vw}) \right] (x_v - x_w),
\]
we can change the above system into matrix format as:
\[
Lx' = b,
\]
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.

**Iteration.** Poisson system basically is an approximation method. The system [ZG16] cannot result in an exact polycube, such as the models with a nearly polycube shape shown in Figure [ZG16]. 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:
\[
L_i x'_i = b_i.
\]

Figure [ZG16] 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.

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.

### 4. Results and Demonstrations

We test our method in a variety of meshes with complicated topology, with or without boundary. Our experiments demonstrate that our algorithms can work on all kinds of shapes.

In Figure [ZG16] we show several models and their polycube shapes. It shows that our algorithms can obtain perfect polycubes no matter the complexity of the model shapes.

The polycubes are also affected by the genus of the models. The technique we propose can manipulate models with high genus directly, such as shown in Figure [ZG16]. All of these models are deformed into their polycubes. Therefore our algorithm is robust to the topologies of meshes.

Our method is insensitive to the boundaries. The figures in [ZG16] 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.](image)

Even on non-orientable meshes, our algorithm can also deform them successfully. In Figure [ZG16] the well-known "costa" surface mesh is deformed into a polycube.

After obtaining the segmentation and the polycube topology, the PolyCut [LVS13] method also presents a deformation algorithm to obtain polycube geometry. Their deformation is based on the vertex normal rotations, our rotations are applied on face normals. Therefore their method and the algorithm in [ZG16] can not process the non-orientable surfaces, and the meshes with boundaries.
The algorithm we present is also robust to the polycube topological defects. When the polycube topology is not valid. Our method can still process the model and output a polycube-like shape with the definitely same polycube topological defects. In Figure 9, the two models with and without topological defects are shown in Figure 9a and 9c. Their corresponding polycube deformation results are displayed in Figure 9b and 9d.

**Comparison.** Many recent algorithms [HJS14] cannot guarantee to obtain a perfect polycube shape without the topological defects except the only one method proposed in [FBL16]. 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 10 and Figure 11.

In speed, our algorithm just solves several linear systems, which is faster than theirs. 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 12 and 13. We can conclude that our algorithm can preserve the edge and area much better than the method in [FBL16].

5. Conclusion and Future work

In this paper, we present a robust, efficient polycube deformation algorithm. Our method is based on the explicitly separating the whole process into three steps. Every step can be processed with dif-
different methods. Our method outperforms previous ones in speed, robustness, simplicity, diversity and quality. Although this deformation technique leads to a direct cross-map between original mesh and its polycube, we cannot guarantee that the map is bijection, and one-to-one. In the future, we plan to contain more constraints in the polycube geometry step to obtain a bijection map.

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.

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 MeshDG [Zha10] framework. We also thank Libigl [JP*16] for reference.

References

[ACOL00] ALEXA M., COHEN-OR D., LEVIN D.: As-rigid-as-possible shape interpolation. In Proceedings of the 27th annual conference on Computer graphics and interactive techniques (2000), ACM Press/Addison-Wesley Publishing Co., pp. 157–164.

[BS08] BOTSCH M., SORKINE O.: On linear variational surface deformation methods. IEEE Transactions on Visualization and Computer Graphics 14, 1 (2008), 213–230.

[CL*10] CHANG C.-C., LIN C.-Y., ET AL.: Texture tiling on 3d models using automatic polycube-maps and wang tiles. Journal of Information Science and Engineering 26, 1 (2010), 291–305.

[CLS16] CHERCHI G., LIVESU M., SCATENI R.: Polycube simplification for coarse layouts of surfaces and volumes. In Computer Graphics Forum (2016), vol. 35, Wiley Online Library, pp. 11–20.

[CPSS10] CHAO L., PINKALL U., SANAN P., SCHRODER P.: A simple geometric model for elastic deformations. ACM Transactions on Graphics 29, 4 (July 2010), 38:1–38:6.

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

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

[FJFS05] FAN Z., JIN X., FENG J., SUN H.: Mesh morphing using polycube-based cross-parameterization. Computer Animation and Virtual Worlds 16, 3-4 (2005), 499–508.

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

[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.

[GBI13] GARCIA I., JIA X., HE Y., XIN S.-Q., PATOW G.: Interactive applications for sketch-based editable polycube map. Visualization and Computer Graphics, IEEE Transactions on 19, 7 (2013), 1158–1171.

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

[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.

[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 (2010), ACM, pp. 127–136.

[HYLG08] HE Y., YIN X., LUO F., GU X.: Harmonic volumetric parameterization using greenâ ˘A ´Zs functions on star shapes. In Symposium on geometry processing, Copenhagen, Denmark (2008).

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

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

[LGW*07] LI X., GUO X., WANG H., HE Y., GU X., QIN H.: Harmonic volumetric mapping for solid modeling applications. In Proceedings of the 2007 ACM symposium on Solid and physical modeling (2007), ACM, pp. 109–120.

[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.

[LVS*13] LIVESU M., VINO R., SHEFFER A., GREGSON J., SCATENI R.: Polycut: Monotone graph-cuts for polycube base-complex models. ACM Transactions on Graphics 32, 3 (July 2013), 25.

Figure 14: The models of high genus and their polycube shapes.
Figure 15: Gallery of our polycube deformations.
construction. Transactions on Graphics (Proc. SIGGRAPH ASIA 2013) 32, 6 (2013). doi:10.1145/2508363.2508388

[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.

[SA07] SORKINE O., ALEXA M.: As-rigid-as-possible surface modeling. In Proceedings of EUROGRAPHICS/ACM SIGGRAPH Symposium on Geometry Processing (2007), pp. 109–116.

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

[WHL*08] WANG H., HE Y., LI X., GU X., QIN H.: Polycube splines. Computer-Aided Design 40, 6 (2008), 721–733.

[WJH*08] WANG H., JIN M., HE Y., GU X., QIN H.: User-controllable polycube map for manifold spline construction. In Proceedings of the 2008 ACM symposium on Solid and physical modeling (2008), ACM, pp. 397–404.

[WYZ*11] WAN S., YIN Z., ZHANG K., ZHANG H., LI X.: A topology-preserving optimization algorithm for polycube mapping. Computers & Graphics 35, 3 (2011), 639–649.

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

[XZW06] XU D., ZHANG H., WANG Q., BAO H.: Poisson shape interpolation. Graphical Models 68, 3 (2006), 268–281.

[YL08] YAO C.-Y., LEE T.-Y.: Adaptive geometry image. Visualization and Computer Graphics, IEEE Transactions on 14, 4 (2008), 948–960.

[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.

[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

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

[ZRKS05] ZAYER R., ROSSL C., KARNI Z., SEIDEL H.-P.: Harmonic guidance for surface deformation. In Computer Graphics Forum (2005), vol. 24, Wiley Online Library, pp. 601–609.