ℓ_1-based Construction of Polycube Maps from Complex Shapes

Jin Huang[1], Tengfei Jiang[1], Zeyun Shi[1], Yiying Tong[2], Hujun Bao[1*], Mathieu Desbrun[3]

[1] State Key Lab of CAD&CG, Zhejiang University, [2] Michigan State University, [3] Caltech

Polycube maps of triangle meshes have proved useful in a wide range of applications including texture mapping and hexahedral mesh generation. However, constructing either fully automatically or with limited user control a low-distortion polycube from a detailed surface remains challenging in practice. We propose a variational method for deforming an input triangle mesh into a polycube shape through minimization of the ℓ_1-norm of the mesh normals, regularized via an as-rigid-as-possible volumetric distortion energy. Unlike previous work, our approach makes no assumption on the orientation, or on the presence of features in the input model. User-guided control over the resulting polycube map is also offered to increase design flexibility. We demonstrate the robustness, efficiency and controllability of our method on a variety of examples, and explore applications in hexahedral remeshing and quadrangulation.

Categories and Subject Descriptors: I.3.5 [**Computer Graphics**]: Computational Geometry and Object Modeling—*Curve, surface, solid, and object representations*

Additional Key Words and Phrases: Polycube map generation, ℓ_1 optimization, sparse normal field

1. INTRODUCTION

Polycube maps were first introduced by [Tarini et al. 2004] for texture mapping purposes. Ever since, it has found applications in a broad range of areas, including parameterization [Yao and Lee 2008; Garcia et al. 2013], reconstruction [Wang et al. 2008], and remeshing [Gregson et al. 2011]. While early approaches required manually-provided topological structures [Tarini et al. 2004], a few automatic polycube construction methods have recently appeared [Lin et al. 2008; Gregson et al. 2011]. However, they often make restrictive assumptions on the initial orientation of the models, rely on prior analysis of the features, or offer no intuitive and effective control for the user. In contrast, we offer in this paper a systematic formulation for the automatic or user-guided construction of polycubes from arbitrary input meshes.

A simple geometric characterization of an axis aligned polycube is that the normal of each of its faces is aligned with one of the axes of an orthonormal coordinate frame. This means that the ℓ_1-norm of every ℓ_2-unit normal vector on its boundary is 1—the minimum among all unit vectors. This key property will allow us to formulate the construction of a polycube as a global minimization. Moreover, we seek low distortion between the input 3D shape and the resulting polycube. We will therefore not only rely on the input triangle mesh deformation, but also on the deformation of an interior tetrahedralization of the input mesh [Tarini et al. 2004; Gregson et al. 2011] to introduce as little distortion as possible in the polycube map between the two volumes. Consequently, and unlike traditional ℓ_1 methods based on convex optimization with linear constraints, we minimize the ℓ_1-norm of a nonlinear function (surface normals) while enforcing that the resulting polycube corresponds to a small volumetric deformation of the input shape. In order to efficiently solve this variational formulation, we use a smooth approximation to the ℓ_1 norm and minimize the global penalty function by a Sequential Quadratic Programming (SQP) based method. This optimization framework also facilitates the inclusion of various controls over the properties of the output, such as distortion, feature preservation, and level of detail.

1.1 Related Work

Polycube maps are relevant to a number of applications, such as cross-parameterization, quadrangulation, and hexahedralization. Since we focus on the automatic and controllable generation of polycube maps, we only discuss the most relevant topics next.

While early polycube map generations heavily relied on human interaction [Tarini et al. 2004; Yao and Lee 2008], recent work such as [Lin et al. 2008] proposed an automated approach to polycube creation through segmentation and fitting of box primitives. Its failure to robustly handle complex models prompted Gregson *et al.* [2011] to introduce a deformation minimization to derive the polycube shape. However, the resulting polycube topology was mostly determined as a preprocessing step through an initial axis alignment of the boundary normals; this heavy dependence of results on the initial orientation limits its ability to deal with complex shapes, and provides only limited room for user interaction.

Hexahedral and quadrilateral remeshing is arguably the most common applications of polycube maps. In recent years, various methods have achieved success in providing high quality controllable quadrangulation from polycubes, even extracting rather coarse quad structure by heuristic quadrilateral simplification [Myles et al. 2010; Tarini et al. 2011] or greedy dual loop selection [Campen et al. 2012]. Unlike quadrangulation, hexahedral remeshing remains challenging. Although the 3D counterpart of a surface cross-frame field has been proposed in, e.g., [Huang et al. 2011], recent progress [Li et al. 2012] shows how non-trivial it is to extend 2D techniques to hexahedral meshing because of the lack of theoretical guarantees on the global topological structure. While a full-blown approach to hexahedral meshing is beyond the scope of this paper, we will demonstrate that applying our polycube map generation approach to this case offers simplicity, robustness and versatility—even if the use of a polycube approximation restricts the topology of the resulting mesh.

[*] Corresponding author: bao@cad.zju.edu.cn

DOI 10.1145/2601097.2601352
http://doi.acm.org/10.1145/2601097.2601352

Typical ℓ_1 norm problems aim at reconstructing sparse signals from a large amount of measurement. It has recently been applied to feature preserving point cloud smoothing [Avron et al. 2010], where the basic idea was to leverage the sparse nature of the normal field of sharp features. Our formulation of the necessary condition for polycube also utilizes the sparsity property of the ℓ_1 norm, albeit in a very different geometric context. Because the ℓ_1 norm of the normal is non-linear in the node position, we adopt existing numerical techniques [El-Attar et al. 1979; Andersen 1996] to turn our ℓ_1 minimization into a smooth optimization problem.

1.2 Overview and Notations

Our approach requires a closed input surface mesh S of arbitrary topology and number of components. We first convert this surface mesh (using, e.g., [Schöberl 1997]) into a tetrahedral mesh $\{T, \bar{X}\}$, where $T = \{t_i\}$ is its set of tetrahedra, and $\bar{X} = \{\bar{x}_i\}$ is its set of node coordinates. The output of our approach is a deformed tetrahedral mesh (possibly with a coarser connectivity) such that the normal of each element of its set $B = \{b_i\}$ of boundary triangles is axis-aligned for an arbitrary orthonormal coordinate system, i.e., a polycube. This output mesh will be found through the constrained minimization of an energy containing terms to enforce axis-alignment of the normal field (described in Section 2) and terms to regularize and control the resulting variational formulation (described in Section 3). A robust and efficient numerical approach to solve this constrained minimization is also introduced (Section 4), and a series of results and applications on challenging models is provided (Section 5).

2. ℓ_1 FORMULATION OF POLYCUBES

We introduce a formulation of an ℓ_1-based energy that will serve as our core component to deform an arbitrary tetrahedral mesh into a polycube-shaped mesh.

2.1 Rationale

For a triangle with ℓ_2-unit normal $\mathbf{n}$, one can define a simple measure of its deviation from being axis-aligned using the ℓ_1-norm $\|\mathbf{n}\|_1 = |\mathbf{n}_x| + |\mathbf{n}_y| + |\mathbf{n}_z|$, through $\|\mathbf{n}\|_1 - 1$. These positive deviations over a given surface made out of triangles $\{b_i\}$ can then be integrated to evaluate how far its shape is from being a polycube, resulting in an energy d of the form:

$$d(\{T, \bar{X}\}) = \sum_{b_i} \mathcal{A}(b_i, X)\|\mathbf{n}(b_i, X)\|_1 - \sum_{b_i} \mathcal{A}(b_i, X), \quad (1)$$

where $\mathcal{A}(b_i, X)$ stands for the area of b_i with the given node position set X. This energy, which is a function of the node positions of the triangles, reaches its zero minimum for a polycube; however its minimization induces spurious shrinking as collapsing all nodes to a point also minimizes this energy. If one replaces all area weighting with the initial areas based on $\bar{X}$, or even removes all area weighting, numerical issues appear as convergence is significantly slowed down and impaired by triangles becoming nearly degenerate. Dividing the above energy by the current total area $\sum_{b_i} \mathcal{A}(b_i, X)$ would partially alleviate these issues, but would also increase its non-linearity, and the resulting dense Hessian matrix makes this alternative energy not amenable to efficient minimization. Finally, one could remedy this issue by rescaling the result at each minimization step; but this procedure leads to very slow convergence.

To render our ℓ_1 formulation well-behaved, yet numerically tractable and efficient, we turn this unconstrained minimization into a constrained minimization: using the ℓ_1-norm of the *area-weighted normal field* and an additional equality constraint to enforce a constant area, we formulate our polycube construction problem as

$$\begin{aligned} &\text{argmin}_X\, E_1(X) \text{ subject to } C(X) = 0, \\ &\text{where } E_1(X) = \sum_{b_i} \mathcal{A}(b_i, X)\|\mathbf{n}(b_i, X)\|_1 \\ &\text{and } C(X) = \sum_{b_i} \mathcal{A}(b_i, X) - \sum_{b_i} \mathcal{A}(b_i, \bar{X}). \end{aligned} \quad (2)$$

With this formulation, E_1 becomes just the ℓ_1-norm of a quadratic function of X with a sparse Hessian, while area shrinkage is prevented by the area constraint $C(X)$.

2.2 Global Orientation

As the orientation of the input is considered arbitrary, we must also find the optimal orthonormal coordinate frame to which to align the resulting polycube. Using bounding boxes or principal component analysis (PCA) to pre-align the input mesh as proposed in previous methods is far from optimal on complex inputs. One may also use an ℓ_2-norm based PCA on the image of the Gauss map rather than on the geometry to improve alignment; however, this approach is unsatisfactory: it is easy to show that even in 2D, such a PCA based method cannot axis-align the two orthogonal edges of a simple right triangle. We instead add a global rotation to our minimization in order to find the best orientation. We simply compute an optimal global rotation matrix R satisfying

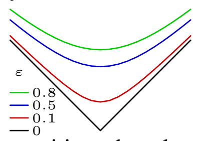
$$\text{argmin}_R \frac{\sum_{b_i} \mathcal{A}(b_i, X)\|R\mathbf{n}(b_i, X)\|_1}{\sum_{b_i} \mathcal{A}(b_i, X)} + \|R^T R - I\|_2^2. \quad (3)$$

This objective function contains an ℓ_1-norm as well, but its low dimensionality (R is a 3×3 matrix) makes it easy to optimize. The mesh is then rotated by the closest rotation matrix to R (through polar decomposition [Irving et al. 2004]) before each step of our constrained minimization of E_1. While the benefit of applying a global rotation decreases as the result gets closer to a polycube, we apply it at each step of our minimization since it only introduces marginal overhead.

2.3 Smooth Approximation of ℓ_1 Norm

Various numerical methods have been proposed to solve ℓ_1-norm problems. Most of them aim at finding a sparse solution to an underdetermined system of linear equations, which turns out to be a convex optimization problem. Our case is rather different, as we are using the ℓ_1 norm of the boundary normals of our mesh with quadratic constraints. To deal with this case, one can use, for instance, interior point methods [Luksan et al. 2007; Schittkowski 2008; Gould et al. 2003]. For simplicity, we instead employ a smooth approximation of the ℓ_1-norm introduced in [El-Attar et al. 1979] (hyperboloid approximation [Andersen 1996]).

Given a component $c \in [-1, 1]$ of the normalized normal, we simply replace the absolute value $|c|$ by $\sqrt{c^2 + \varepsilon} := \tilde{c}$ where $\varepsilon \geq 0$ is a regularizing parameter to balance smoothness and accuracy. The inset plot compares a few different values of ε: small values better capture the rapid change of slope near zero at the risk of instability during optimization, while large values produce smoother transitions, but also slow down convergence. As proposed in [El-Attar et al. 1979], we



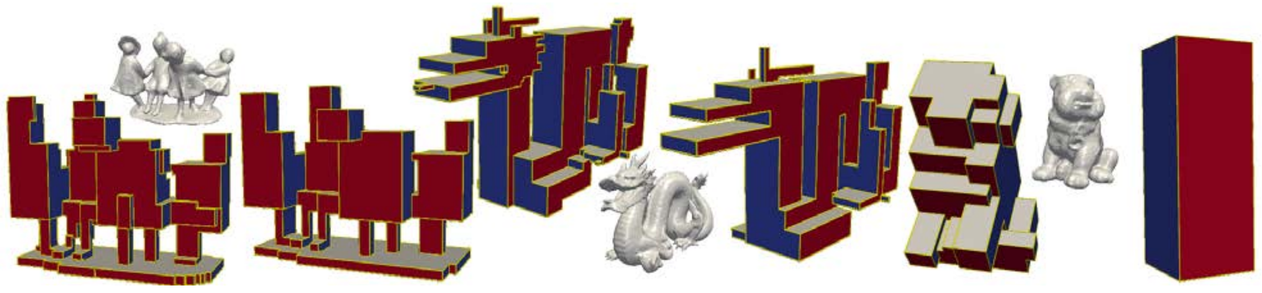


Fig. 1. Through our ℓ_1 minimization of the boundary normal field of the tetrahedral mesh of a shape, a low-distortion polycube shape emerges automatically. The user can easily control the complexity of the resulting polycube structure (two polycube maps are shown for each input). The resulting polycube map facilitates a series of geometry processing tasks such as texturing or remeshing of the input model. (Models "dancing children" ©IMATI-GE, "bulldog" ©VCG-ISTI, "dragon" ©Stanford 3D Scanning Repository.)

start from a relative large ε and decrease it gradually throughout our minimization procedure. (Another common alternative is to use Huber's M-estimator function [Huber 1981; Pinar and Hartmann 2006]; it led to slightly slower convergence in our experiments.) Note that this smoothed version is numerically preferable to a naive optimization which would "snap" boundary normals to their closest axes at each iteration: while the latter creates jittering and instability during optimization, our approach still captures the ℓ_1 nature of the problem while offering a smooth energy landscape.

The gradient (with respect to node positions) of this approximate absolute value is expressed as a function of the original component:

$$\nabla \tilde{c} = \frac{c}{\tilde{c}} \nabla c. \tag{4}$$

The Hessian $\nabla^2 \tilde{c}$ consists of two terms:

$$\frac{\varepsilon}{\tilde{c}^3} \nabla c \nabla c^T + \frac{c}{\tilde{c}} \nabla^2 c. \tag{5}$$

Note that the second term is not necessarily semi-positive-definite; we thus approximate it by the closest SPD Hessian matrix to increase numerical stability, which can be evaluated by clamping the eigenvalues of $\nabla^2 \tilde{c}$ to non-negative values. This mixed gradient-Hessian approximation provides an accurate gradient evaluation and an efficient symmetric semi-positive-definite Hessian approximation of our ℓ_1-based energy, two key ingredients for the Sequential Quadratic Programming (SQP) solver we will describe in Section 4.

3. REGULARIZATION AND CONTROLS

So far, we have only provided an energy whose minimization guarantees that the input shape turns into a polycube. We now need to add volumetric regularization and optional user-guided controls to single out the desired polycube shape from the large set of stationary points of Eq. (2).

3.1 Low-distortion Polycubes through Regularization

It is desirable for a polycube map to be of low distortion, i.e., the deformation from the input shape to its polycube approximation should be as small as possible. This constraint is in general not just a statement about surface deformation, but also about volume deformation. While the polycube energy relied only on the boundary triangles of the shape thus far, we now measure distortion using the entire volumetric tetrahedral mesh $\{T, X\}$ as in [Gregson et al. 2011]. We adopt the "as-rigid-as-possible" (ARAP) distortion measurement [Alexa et al. 2000]: for each tet t_i, we measure:

$$\delta(t_i, X) = \frac{1}{2} \|G_i X - \mathrm{polar}(G_i X)\|_F^2, \tag{6}$$

where G_i is the operator which, applied to X, returns the 3×3 gradient matrix for the current shape of t_i with respect to the undeformed shape $\bar{X}$, polar(.) returns the closest rotation matrix through polar decomposition [Irving et al. 2004], and $\|\cdot\|_F$ denotes the Frobenius norm. As often done in the literature, we approximate the gradient and Hessian of the distortion by $G_i^T G_i X - G_i^T \mathrm{polar}(G_i X)$ and $G_i^T G_i$ respectively.

The normalized total distortion energy is then defined as the integral of these tet-based distortion measurements over the input shape:

$$E_\delta(X) = \frac{\sum_{t_i} \mathcal{V}(t_i, \bar{X}) \delta(t_i, X)}{\sum_{t_i} \mathcal{V}(t_i, \bar{X})}, \tag{7}$$

where $\mathcal{V}(t_i, \bar{X})$ stands for the volume of the element t_i for the original node positions $\bar{X}$. This energy is added to the ℓ_1-based polycube energy in order to regularize it and penalize any large distortion, ensuring that our resulting polycube has only limited volumetric deformation. Note that for applications of polycube maps where volumetric distortion is of little inconvenience (such as quadrangulation and parameterization), one can replace this volumetric definition by a surface distortion penalty instead (see Section 5.3).

3.2 Incorporating User-guided Controls

Combining the ℓ_1-based energy E_1 and the distortion penalty E_δ provides a variational formulation for the construction of polycube maps. However, user control is desirable as well: one often needs added flexibility in the design of polycube through intuitive editing of the results. Our energy-based formulation allows the addition of user-guided penalty terms to offer design control.

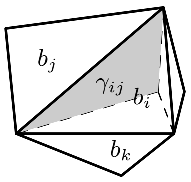
Controlling Shape Complexity. One typical design need is to control the amount of detail preservation in different regions of the polycube: one often wishes to locally allow for slightly larger distortion if this greatly simplifies the polycube shape (see Fig. 1). Our ℓ_1 energy is, however, blind to the number of stairs in the final polycube. We thus need to incorporate an additional term to control the final shape complexity. A simple idea is to define a measure η of polycube complexity by summing the normal differences between all adjacent boundary triangle pairs (b_i, b_j) through:

$$\eta(b_i, b_j, X) = \beta \mathcal{A}(e_{ij}, X) \left\| \mathbf{n}(b_i, X) - \mathbf{n}(b_j, X) \right\|_2^2, \quad (8)$$

where $\mathcal{A}(e_{ij}, X)$ is a local area associated to edge e_{ij} between b_i and b_j, and β is a scalar indicating the strength of this penalty. Because the normalized normal is numerically unstable for degenerate triangles, we define $\mathcal{A}(e_{ij}, X) = \gamma_{ij}(X)\mathcal{A}(b_i, X) + \gamma_{ji}(X)\mathcal{A}(b_j, X)$ to reduce the contribution of the above measurement around nearly degenerated triangles:

$$\gamma_{ij}(X) = \frac{\mathcal{A}(b_j, X)}{\sum_{b_k \in \mathcal{N}(b_i)} \mathcal{A}(b_k, X)}, \quad (9)$$

where $\mathcal{N}(b_i)$ is the three neighboring triangles of b_i. As shown in the inset, the weights $\{\gamma_{ij}\}$ split the area of a triangle into three parts, each being proportional to the area of the adjacent triangle. Thus $\mathcal{A}(e_{ij}, X)$ will be close to zero if its adjacent triangle is degenerate, while keeping the sum of all edge areas equal to the total boundary area. Note that we can safely discard the derivative of $\mathcal{A}(e_{ij}, X)$ for simplicity as triangle areas change smoothly during the optimization. The complexity penalty $E_\eta(X)$ is finally expressed as



$$E_\eta(X) = \sum_{b_i, b_j \,\text{adjacent}} \eta(b_i, b_j, X) \;/\; \sum_{b_k} \mathcal{A}(b_k, \bar{X}), \quad (10)$$

where we normalized the edge-weighted normal differences by the input boundary area. We apply the same numerical treatment as the one used in the ℓ_1 energy to get a semi-positive-definite Hessian approximation. Finally, we add the inequality constraints

$$D(X) = \mathbf{n}(b_i, X) \cdot \mathbf{n}(b_j, X) + 1 > 0, \quad (11)$$

to prevent the spurious case of adjacent normals becoming opposite (i.e., the creation of foldovers) through a log barrier method within our SQP interior point solver. As a result, our complexity penalty and constraints offer a reliable means to control the complexity of our resulting polycube maps (see Fig. 17).

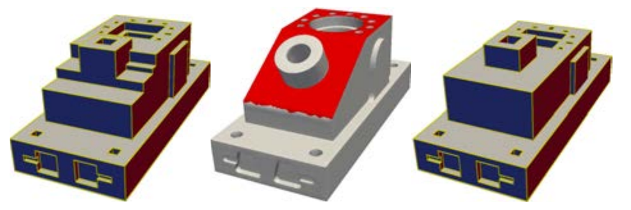


Fig. 2. The user can optionally select regions (red) to guide polycube edges, thereby controlling the details of the resulting polycube map more accurately (left: ω=0, right: ω=10^3). (Model "anc101" ©INRIA.)

Controlling Flat Regions and Sharp Edges. The user may sometimes want to guide the placement of the polycube edges: having polycube edges at specific sharp features may be particularly convenient for, e.g., texture packing; conversely, flat or nearly flat regions may be better placed on faces of the polycube. We therefore add a final energy $E_e(X)$, in which user-defined edges between triangles b_i and b_j (selected either via mouse interaction or automatic edge extraction) that are desirable to have as part of the edge skeleton of the final polycube are penalized via an ℓ_2-based energy of the form:

$$\omega[\mathbf{n}(b_i, X) \cdot \mathbf{n}(b_j, X)]^2. \quad (12)$$

where ω is a scalar value indicating how forcefully the user needs this edge to be at a right dihedral angle in the resulting polycube. Similarly, user-selected edges that should end up being flat in the polycube are handled by adding instead an ℓ_2-based energy of the form:

$$\omega[(\mathbf{n}(b_i, X) \cdot \mathbf{n}(b_j, X) - 1]^2, \quad (13)$$

The final edge-feature control energy $E_e(X)$ is simply the sum of all these edge-based optional terms. This treatment of edges allows for a very intuitive editing of the polycube, should the current result not suit the user (see Fig. 2 comparing two values of ω over a chosen region of the input; note that we did not use this term for any other result shown in this paper).

4. NUMERICS AND EXTRACTION

We now describe how we numerically solve the resulting constrained minimization of the total energy obtained by simply summing the four aforementioned energies:

$$\begin{aligned} &\underset{X}{\operatorname{argmin}} \; \alpha E_1(X) + E_\delta(X) + E_\eta(X) + E_e(X) \\ &\quad s.t. \quad C(X) = 0, \text{ and } D(X) > 0, \end{aligned} \quad (14)$$

where E_1, E_δ, E_η, and E_e are the energies derived from respectively the ℓ_1-based polycube constraint, and the ℓ_2-based regularization, complexity, and edge-feature penalties. $C(X)$ is the total area constraint (Eq. (2)), and $D(X)$ is the anti-foldover constraints (described in Eq. (11)) implemented via a log barrier. The weight α is automatically adjusted based on a scheduled strategy as explained below.

Basic Optimization Procedure. Directly applying Lagrange-Newton method to solve the above problem involves the Hessian of the constraint $C(X)$, which may not be semi-positive-definite. We thus use a modified Lagrange-Newton method that simply discards the Hessian of the constraints. Moreover, we adopt the Gauss-Newton scheme to approximate the gradients and Hessians of the ℓ_2-based energies. In each iteration, we compute the increment ΔX of X by solving the linear system:

$$\begin{pmatrix} H(X) & \nabla C(X) \\ \nabla C(X)^T & 0 \end{pmatrix} \begin{pmatrix} \Delta X \\ \lambda \end{pmatrix} = \begin{pmatrix} -g(X) \\ -C(X) \end{pmatrix} \quad (15)$$

where $g(X)$ and $H(X)$ denote the approximated gradient and Hessian of the total energy. The matrix above is positive semi-definite by design, but the gradient of the constraint ∇C is still rather dense for boundary nodes. We therefore efficiently solve this linear system using the Schur complement method [Benzi et al. 2005] by sequentially evaluating:

$$\begin{aligned} \lambda &= \frac{-\nabla C(X)^T H(X)^{-1} g(X) + C(X)}{\nabla C(X)^T H(X)^{-1} \nabla C(X)} \\ \Delta X &= -H(X)^{-1} g(X) - \lambda H(X)^{-1} \nabla C(X). \end{aligned} \quad (16)$$

Multiplication by the inverse of $H(X)$ is computed via Cholesky factorization of H [Chen et al. 2008] followed by back-substitution. During the optimization, if the energy cannot further decrease because of the log barrier penalty (i.e., if we detect a large gradient, but are required to take a small step to avoid foldovers), we collapse slim boundary triangles (simultaneously on both the input mesh and the current deformed one) that contain angles less than 5° before resuming the optimization. This treatment safely and efficiently avoids numerical slowdowns due to near degenerate triangles.

Robust Weighting Schedule. Obviously, boundary triangle normals will become axis-aligned only if a large weight α for the ℓ_1-norm term is used. However, starting with such a large weight basically removes the effect of our regularization, and thus often leads to the minimization procedure getting quickly trapped in a local minimum. Thus, we introduce a simple and robust weighting schedule. We begin by solving the constrained minimization (Eq. (14)) with $\alpha = 0.1, \varepsilon = 1$. We assume convergence of the minimization procedure when the polycube error $d(\{T, \bar{X}\})$ provided in Eq. (1) stops decreasing for five consecutive iterations: more precisely, if the initial polycube error of the current minimization procedure is d_0, we track the progress of the i-th solve of Eq. (15) during the minimization by computing its current polycube error d_i and by recording the change of error $\Delta d_i = |d_i - d_{i-1}|$; convergence is declared if the sum of the error changes over the last five iterations is smaller than $d_0/100$ (i.e., no real progress is being made), or if the SQP converges. We then double the value of α, decrease ε by a half if it is larger than 0.01, and start a new constrained minimization procedure, initialized with the current solution, and run to convergence. In our experiments, the final polycube error $d(\{T, \bar{X}\})$ often does not decrease significantly after 20 iterations; additionally, we can robustly extract a polycube as soon as the (normalized) error is smaller than 0.001. Thus we repeat this doubling of α at most 20 times or until the error is smaller than 0.001. Fig. 3 shows the typical evolution of error in our solver, where each doubling of α and decrease of ε kicks in when the error bottoms out. Notice the monotonically-decreasing behavior of the energies involved in our optimization. We tried various other schedules as shown in Fig. 4: starting from very large $\alpha = 10$, small $\varepsilon = 0.1$ and using fast updates (multiply and divide by 4 respectively) gets the results stuck in local minima and leads to highly distorted shapes, while beginning with very small $\alpha = 0.001$, large $\varepsilon = 10$ or using slow updates (multiply and divide by $\sqrt{2}$ respectively) makes the convergence slow. Our choice achieves a good balance between quality and performance, and works well for all the results shown in the paper as we will further demonstrate in Section 5.

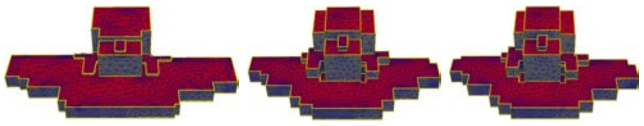

(a) 3/46/0.150 (b) 9/88/0.0819 (c) 31/226/0.0789

Fig. 4. Results for angel model with different schedules: (a) large α and small ϵ with large updates, (b) our preferred schedule, (c) small α and initial ϵ. Bottom row indicates the number of α & ε updates/number of solves/final E_δ. Schedule (b) offers a good balance between efficiency and quality. (Model "angel" ©INRIA.)

4.1 Polycube Cleanup

Once the polycube optimization is complete, we label surface triangles by patches according to their closest axis of surface normal (i.e., based on the largest component of the normal) as proposed in [Gregson et al. 2011]. We then "straighten" the chart border by relabeling triangles along the border to remove potential zigzags; e.g., if a surface triangle is labeled x, but has three neighbors labeled x, y, and y, we relabel it as y. We then build the incidence graph of patches to detect topological degeneracies. If we find a patch of degree one (i.e., an isolated patch), we relabel its triangles with the surrounding patch type. If a patch is of degree two (i.e., a wedge patch), we relabel its triangles to the type of the neighboring patch that shares the longest boundary with it. Degree-three patches can also be cleaned up; but in practice, we found that iteratively removing degree-one and -two patches avoids degree-three patches altogether. Fig. 5 shows this procedure on the elephant model. This topological cleanup suffices in all the cases we tried. Once labeling and cleanup is complete, the boundary tetrahedral mesh has a proper polycube topological structure. We note here that the other cleanup rules proposed in [Gregson et al. 2011] (multi-orientation splitting and geometry wedge removal) were never necessary in all our tests due to the quality of our results, even for very complex and high genus models as in Fig. 6.

If the interior polycube map is required, we then run a final minimization using only E_δ to deform the original tetrahedral mesh based on the final labeling of normal directions for robustness. Given the input model M (potentially coarsened during optimization), we deform it under the linear constraints on its boundary ∂M that the nodes in a same patch share the same coordinate in the normal direction. To further improve the robustness, we solve the following linearly constrained quadratic optimization to relax the node position according to its valence, and then use it as the initial value for the minimization of the ARAP-based E_δ:

$$\frac{1}{|\partial M|}\sum_{i\in\partial M}\left\|x_i - \frac{\sum_{j\in\mathcal{N}(i)\cap\partial M} x_j}{|\mathcal{N}(i)\cap\partial M|}\right\|_2^2 + \frac{1}{|M\backslash\partial M|}\sum_{i\notin\partial M}\left\|x_i - \frac{\sum_{j\in\mathcal{N}(i)} x_j}{|\mathcal{N}(i)|}\right\|_2^2, \quad (17)$$

where $\mathcal{N}(i)$ is the node indices of the one-ring neighborhood of the node i. This helps improve the smoothness of the interior polycube map at very low computational cost, since this constrained optimization is easily turned into an unconstrained one through variable substitution.

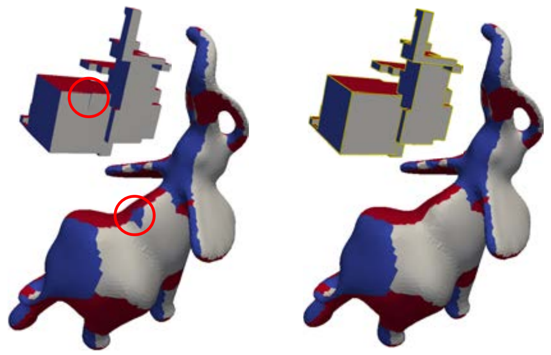


Fig. 5. While the polycube found through optimization may contain a few spurious topological degeneracies (left), a straightforward cleanup of the adjacency graph between resulting polycube faces reliably removes these issues. Here we stopped the solve after 6 α-doubling steps to get a clear degeneracy (left). The final polycube with correct topology is obtained after cleanup (right). (Model "Livingstone elephant" ©INRIA.)

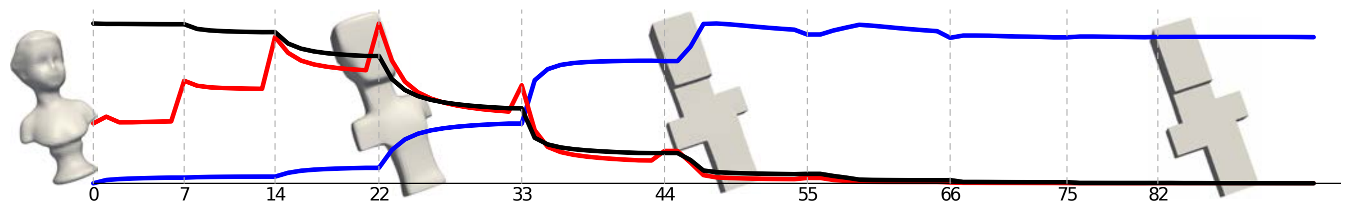


Fig. 3. Optimization of the "buste" model. Starting with an initial polycube quality of 0.483632, the optimization stops after 89 iterations and a total of 9 doubling steps of α (each doubling is indicated by a vertical grey line). The shapes show the results after the 3th, 5th, 9th change of α. For legibility, we linearly stretched the values (red for E_1 , blue for E_δ and black for the polycube error defined in Equation (1)) of the plot to map between 0 and 1. (Model "buste" ©Utrecht Uninversity.)

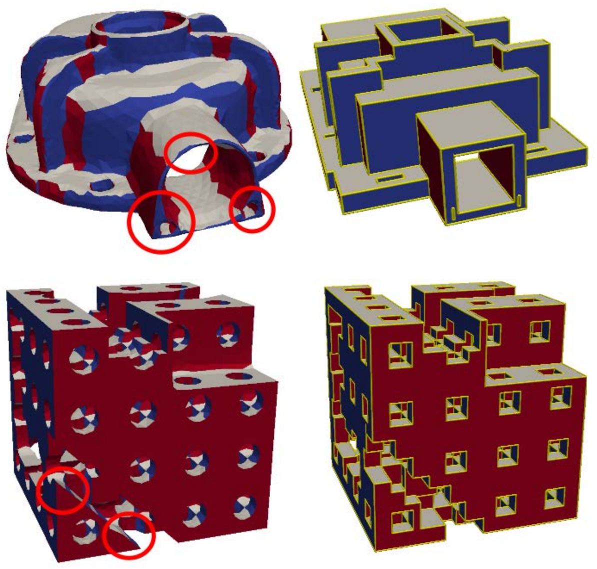


Fig. 6. Even for input models with very sharp edges (left; notice the circled wedges with large dihedral angle), our ℓ_1-based optimization successfully returns high quality polycubes. (Models "cognit, nastychees" ©INRIA.)

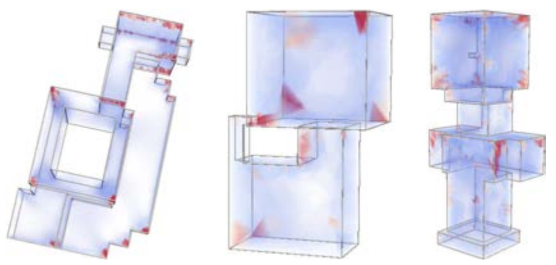


Fig. 7. Our polycube maps have low distortion throughout the volume as evidenced by a volumetric rendering of the as-rigid-as-possible distortion (Eq. (6)) of each tetrahedron. (Models "rockerArm" ©INRIA, "kitten" ©Frank ter Haar.)

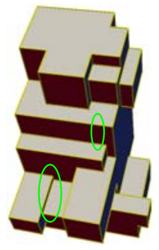
5. RESULTS AND APPLICATIONS

We now present the results of our method for a variety of input meshes ranging from natural shapes to computer-aided design models. We also demonstrate the benefits of our polycube construction on two important applications: hexahedral remeshing, and surface quadrangulation.

Fig. 7 shows visualizations of distortion through volumetric rendering. As expected, the most distorted regions (red) are located near the surface, especially near the polycube edges and corners; internal elements are smoothly deformed with low distortion (blue). Another key insight on the behavior of our method can be seen from a typical convergence plot as in Fig. 3: one can see the ℓ_1-based energy E_1 (in red) steadily decreasing throughout the optimization, while the ARAP distortion E_δ (in blue) progressively and smoothly increases. We also point out that that ℓ_1-based polycube error defined in Eq. (1) (in black) provides a very reliable way to decide when to double α as its graph behavior is quite monotonous (we tried several other measurements which, because of the nonlinearity of our energies, turned out to be unreliable). In Fig. 8 we map the x, y, z components of boundary triangle normals to r, g, b color channels respectively, and highlighted the boundary edges between adjacent triangles with different orientations. As the optimization goes, the normal smoothly changes and the topological complexity (depicted by the highlighted edges) reduces. Besides visual inspection, the quality of a polycube map can be also measured by the ARAP distortion energy [Alexa et al. 2000] of the tetrahedron elements inside the polycube shape. We provide statistics about the quality and performance of a series of models in Tab. II. Timing was measured on a computer with Intel Core i7-3770 processor and 16GB memory. Even if the minimization does not fully converge in 20 α-doubling steps (e.g., for the anc101 model in Fig. 2), the final polycube structure can be easily extracted from the resulting shape.

5.1 Controllability

We also tested our complexity control on different models to evaluate how intuitive the complexity parameter β is. Each of our energies being normalized, this parameter is quite easy to tune to get the right amount of details: Fig. 2 and Fig. 8 show two typical effects of tuning β on large input meshes. If small and localized details remain (see Section 3.2), the user can paint-brush edge constraints to simplify the result further, giving the user efficient control on the final polycube (see the inset which further simplifies the polycube shown in Fig. 1). Note that, except for this illustrative example and Fig 2, no results shown in the paper were manually modified after the topological cleanup stage.



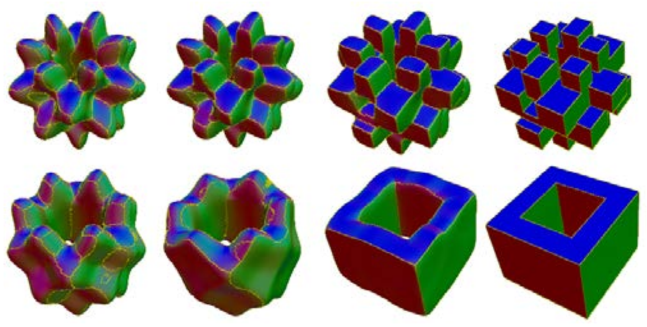


Fig. 8. Our method progressively deforms a model via constrained optimization into a polycube shape. The smoothed formulation of the ℓ_1 norm of the boundary normals makes the final, optimal face orientation emerge from our distortion minimization. Various controls can be applied, e.g., detail preservation (top $\beta = 2$, bottom $\beta = 200$): as the optimization proceeds, our penalty of adjacent normal differences removes small stairs (compare top and bottom results) and the shape converges towards a simpler final polycube.(Model "bumpy torus" ©Max Planck Institute for Computer Science.)

5.2 Hexahedral Remeshing

Since our approach offers a robust polycube construction from arbitrary surface meshes or volume meshes, we can further modify it to remesh a domain with hexahedral elements only. Hexahedral remeshing only imposes one additional requirement over a polycube map: the coordinates of the corners should be integers so that we can easily extract hexahedra from its volume. Consequently, we modify the last step of our polycube generation procedure by applying an integer snapping to meet this requirement: in order to guarantee no degeneracy, we sort the patches that are orthogonal to each axis direction, and snap their position to an integer one by one.

Visual and numerical evaluations of our hexahedral remeshing results can be found in Fig. 14, Fig. 15, Fig. 17 and Tab. II (we use scaled Jacobian to measure the quality of the hexahedral elements, and Hausdorff distance to measure how close the mesh is to the original shape).

5.3 Surface Quadrangulation

As mentioned in Section 3, replacing the interior distortion regularization by a surface distortion measurement suffices to obtain a polycube map relevant for quadrangulation—see Fig. 9. We adopt the ARAP-based distortion measurement [Botsch et al. 2006] in the following results. In fact, the topological structure of the resulting polycubes can be made even simpler without inducing large distortion, and of course, more efficiently. As a result, after carefully grouping the integer variables without introducing degenerated quad patches, our method is able to produce high quality quadrangulations (see Tab. I), be they fine or very coarse. Note that our method can also be applied to locate cone singularities in multiples of $\pi/2$ with distortion control. Finally, as shown in the inset, our method can even be directly applied for models with open boundary with an additional ℓ_1 norm on boundary edges. (Model "venus" ©INRIA.)

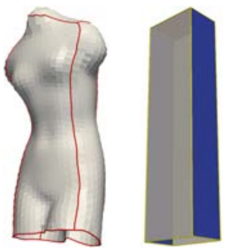


Table I. Quadrilateral mesh quality

model	#quad	#corner	Scaled Jac.	Hausdorff
fandisk	724	38	0.239/0.620/0.999	3.82
elk	6654	68	0.292/0.646/0.999	2.33
casting	8074	144	0.128/0.564/0.999	1.99
rockerArm	8836	57	0.419/0.710/0.999	1.58
kitten	2284	20	0.464/0.732/0.999	2.92
venus-hole	2610	8	0.526/0.763/0.999	0.512

Quadrilateral mesh quality using our polycube maps (the Hausdorff distance ($.10^{-3}$) is normalized by the bounding-box diagonal span of the initial model for comparison purposes).

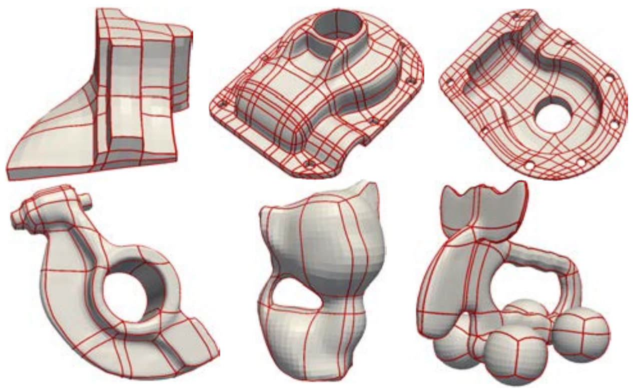


Fig. 9. Coarse quadrilateral domain (highlighted by the red lines) of input meshes can be generated by applying our method without volumetric distortion penalty. (Models "fandisk, elk" ©Max Planck Institute for Computer Science , "casting" ©INRIA.)

5.4 Comparisons

Very few methods are offering a fully automatic or user controlled polycube generation. We compare our approach to these state-of-the-art techniques, both in terms of raw quality and in terms of controllability.

Slicing. A first automatic approach by He et al. [2009] slices the input object into layers in the gradient direction of a harmonic function in order to find its associated polycube. As the harmonic function is constructed from a pair of bottom-most and top-most points on the mesh, the initial orientation of the input is crucial to the quality of the output (see Fig. 11). In each slice, the model is approximated by a quad-tree, which tends to make the resulting polycube an overly voxelized version of the input model depending on the user-defined number of slices: large separations between slices introduce large distortion, while small ones lead to staircase effects as illustrated in Fig. 10. In contrast, our approach captures the symmetries of the input, even without any user input. Fig. 11 further illustrates the artifacts of the slicing approach (dependence on initial orientation and slice distances), compared to our technique which automatically orientates the model and controls the details of the output polycube shape based on the induced distortion. While our method can be notably slower (up to a factor eight in our tests), it achieves significantly better results in reasonable time: for a sphere model with 10k triangles, it took 0.1 and 0.5 minutes respectively for [He et al. 2009]'s method to construct the two polycubes in Fig. 10; our method took 1.3 minutes (59 iterations). For the RockerArm model in Fig. 11 (24k triangles), it took

0.5 and 1.0 minutes for [He et al. 2009]'s method, while it took 6 minutes (86 iterations) for ours (displayed on Fig. 17) .

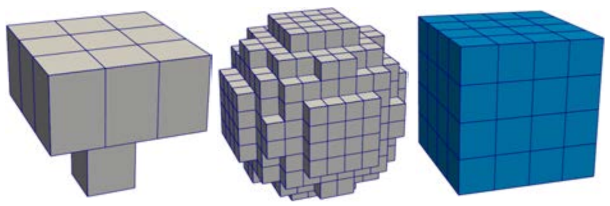


Fig. 10. Sphere model: left and middle are polycubes with different distances between slices using [He et al. 2009]; right is our result.

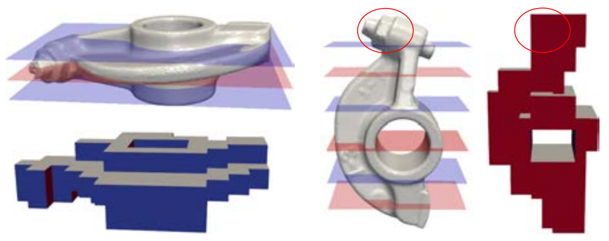


Fig. 11. Rocker arm: using [He et al. 2009], the number of slices depends heavily on orientation of the input model, and details cannot be locally captured (red circle). Instead, our method is orientation independent, and detail control is enforced as a by-product of distortion minimization, leading to better global structures (see Fig. 17 for comparison).

Normal Snapping. The method proposed by Gregson et. al [2011] starts with a model with a prescribed orientation. Their method iteratively snaps the surface normal to an axis direction based on a volumetric deformation gradient. Here again, our approach fares better due to its input orientation independence: as shown in Fig. 12, starting from two different orientations, our method leads to the same polycube topological structure, while their technique fails to do so. Moreover, their process does not go all the way to normals aligned with the axes: as the mesh is deformed by integrating a deformation gradient, the final shape is far from being polycube, which requires significant cleanup to remove topological issues (Fig. 13). Finally, our method achieves better quality in the output hexahedral mesh due to our distortion-based strategy for boundary normal adjustment (see Fig. 14, Fig. 15, Fig. 16). Our ability to easily edit the results is also a notable difference with their work. Note finally that the high quality and robustness of our results come at an extra cost in computational time: for the bulldog model in Fig. 12 (34k tets), our method took 5.7 minutes (108 iterations), while the rotation-driven deformation of [Gregson et al. 2011] only took 1.6 minutes.

5.5 Limitations

Although our penalty energy on normal differences leads to drastically reduced shrinkage compared to the Laplace-based penalty of node positions, too large a weighting of this penalty may cause degeneracy in thin regions as demonstrated in the inset.

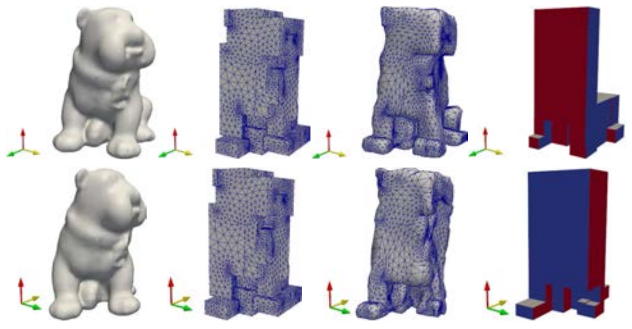


Fig. 12. From left to right: the original model, our results before cleanup, the results from [Gregson et al. 2011] before and after cleanup. The results of our method for two different initial orientations (top vs. bottom) have the same polycube shape (ARAP distortion is 0.0792 before cleanup), while the results of [Gregson et al. 2011] are sensitive to input orientation, and have very different ARAP distortions before clean-up (resp., 0.1287 and 0.1897).

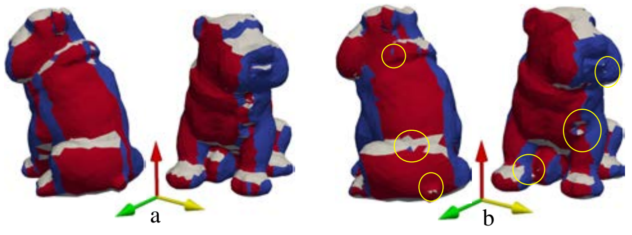


Fig. 13. Topological clean-up: for the example shown in Fig. 12 our method (a) achieves a nearly perfect axis-aligned polycube shape, thus requiring very little topological cleanup (only 1 boundary zigzag and 2 patch degree issues in this case). Instead, [Gregson et al. 2011] (b) requires cleanup for 3 boundary zigzags and 15 patch degree issues.

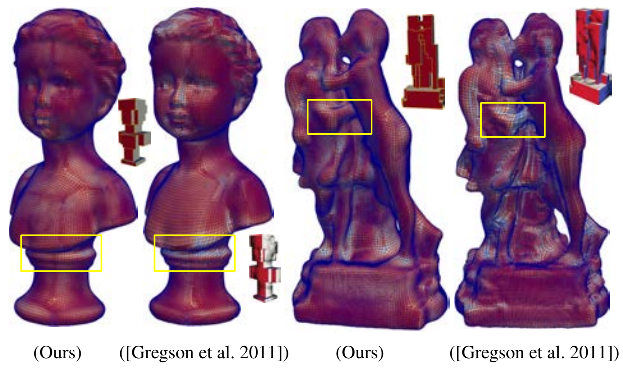


Fig. 14. For a similar complexity of the resulting polycube, our method generates higher quality hex meshes with lower distortion compared to [Gregson et al. 2011]. (Model "kiss" ©INRIA.)

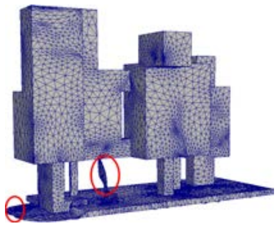

This is, however, *not* a failure of the anti-foldover penalty of Eq. (11): there are in fact several degenerated triangles between patches with opposite normal directions, but removing these triangles by our current adaptive remeshing method leads to invalid tetrahedral mesh connectiv-

Table II. Quality and performance statistics.

model	#tet	$\#\alpha$	β	#iteration	time(m)	ARAP	#corner	#hex	Scaled Jac.	Hausdorff ($\times 10^{-3}$)
angel-1	21036	9	0.1	88	2	0.0819	121	14068	0.470/0.923/0.999	1.59
angel-2	21036	11	1	106	3	0.0978	56	15690	0.212/0.919/0.999	2.81
angel-3	21036	9	4	92	2	0.0801	36	14336	0.222/0.898/0.999	1.35
anc101	155078	20	0.1	87	45	0.0100	248	62858	0.150/0.964/0.999	0.272
bumpy torus*	-	-		-	-	-	206	35137	0.270/0.891/0.999	7.86
bumpy torus	110942	9	2	109	25.5	0.0792	222	38665	0.335/0.929/0.999	1.13
bunny*	-	-		-	-	-	90	81637	0.138/0.930/0.999	4.01
bunny	169341	15	4	134	130	0.0688	76	37734	0.382/0.926/0.999	3.02
fertility*	-	-		-	-	-	96	19870	0.196/0.911/0.999	6.96
fertility	90792	14	1	108	28.6	0.0362	96	17910	0.312/0.911/0.999	0.797
buste*	-	-		-	-	-	88	193664	0.235/0.925/0.999	13.2
buste	94261	14	1	85	28.4	0.00129	93	208928	0.302/0.934/0.999	0.659
kiss*	-	-		-	-	-	153	215320	0.125/0.919/0.999	10.2
kiss	142569	18	1	102	37.3	0.00109	160	219009	0.247/0.921/0.999	0.543
bulldog	34204	9	0.2	108	5.7	0.0884	131	47915	0.215/0.916/0.999	1.06
cognit	29491	9	0.2	197	12.9	0.0864	200	77559	0.270/0.829/0.999	0.643
dancing children-1	117684	14	0.2	138	20	0.0896	360	-	-	-
dancing children-2	117684	18	2	140	27.7	0.123	210	35293	0.143/0.870/0.999	1.17
dragon	106177	9	0.1	118	13.5	0.0929	512	-	-	-
dragon	106177	11	0.5	144	28.1	0.0952	315	117725	0.150/0.857/0.999	0.913
livingstone elephant	132796	9	1	160	23.4	0.0924	160	171657	0.221/0.878/0.999	0.531
gargoyle	213550	11	1	128	110	0.0777	232	25669	0.196/0.906/0.999	3.39
kitten	4951	9	1	97	0.6	0.1021	30	7083	0.424/0.910/0.999	1.61
rockerArm-1	127746	9	1	110	39.5	0.0575	88	24346	0.439/0.920/0.999	1.80
rockerArm-2	127746	11	4	127	30.6	0.0809	64	24780	0.378/0.905/0.999	2.03
rod	79936	9	1	101	48	0.0653	32	11092	0.418/0.929/0.999	0.482
nastycheese	18174	9	0.1	96	8.9	0.0441	970	-	-	-

The columns correspond to respectively the input name, the number of input tets, the total number of α-doubling steps, the value of β for complexity control, the total number iterations, the total time excluding post processing, the average ARAP energy E_δ, and the number of corners of the polycube as an indication of its complexity. We also indicate, for our hexahedral meshes created with the resulting polycubes, the number of elements, the scaled Jacobian (min/mean/max) (mesaured with VERDICT library [Sandia 2010]), and the Hausdorff distance ($.10^{-3}$) (measured with M.E.S.H tool [Aspert et al. 2002]) to the input normalized by its bounding-box diagonal span. The rows marked with "*" are quality statistics from [Gregson et al. 2011].

ity. These issues are easily addressed in practice through a spatially-varying weighting of normal differences, and a better adaptive remeshing techniques; however, we do not know of a way to guarantee success for arbitrary shapes, even if all complex shapes we tried are dealt with remarkably well.

6. CONCLUSION

We presented a novel formulation for polycube construction. A robust numerical procedure is devised to quickly converge to a polycube map by iteratively minimizing the ℓ_1 norm of the normal field while ensuring a low volume distortion and local, optional user guidance. Besides demonstrating the quality of our results on a variety of challenging models, we also explored the use of these polycubes in all-hexahedral mesh and quadrangular mesh generation. For future work, we point out that our approach could be used for the design of orthogonal-symmetric direction field. Comparing our results to the recent work of Knöppel et al. [2013] on the construction of a globally optimal direction field would be informative, and maybe we could even incorporate their technique to alleviate the risk of getting trapped in local minima. A flexible and robust method to easily design hexahedral meshes with control over the topological singularities would also be a valuable extension. Finally, while our approach has excellent numerical robustness even for very challenging models, we suffer from the same lack of rigorous guarantees of convergence and non-degeneracy as all recent works do.

Acknowledgements

We would like to thank the anonymous reviewers for their valuable comments and suggestions. Special thanks to Ying He and James Gregson for providing help on making comparison. This work was partially supported by NSFC (No.61210007, No.61170139), China 863 Program (No. 2012AA011503) and the Fundamental Research Funds for the Central Universities (No. 2013FZA5015). Yiying Tong was supported by NSF (CMMI-1250261 and IIS-0953096). Mathieu Desbrun was supported by NSF grant CCF-1011944.

Fertility and buste are courtesy of Utrecht University. Dancing children is courtesy of IMATI-GE. Bulldog and gargoyle are courtesy of VCG-ISTI. Kitten is courtesy of Frank ter Haar. Venus, rockerArm, casting and livingstone elephant are courtesy of INRIA. Fandisk, elk, bumpy torus and bunny are courtesy of Max Planck Institutfr Institute for Computer Science. Rod, kiss, angel, anc101, nastycheese and cognit are provided by INRIA Gamma dataset. All the other models are provided by the AIM@SHAPE Shape Repository.

REFERENCES

ALEXA, M., COHEN-OR, D., AND LEVIN, D. 2000. As-rigid-as-possible shape interpolation. In *ACM SIGGRAPH*. 157–164.

ANDERSEN, K. D. 1996. An efficient newton barrier method for minimizing a sum of euclidean norms. *SIAM Journal on Optimization 6,* 1, 74–95.

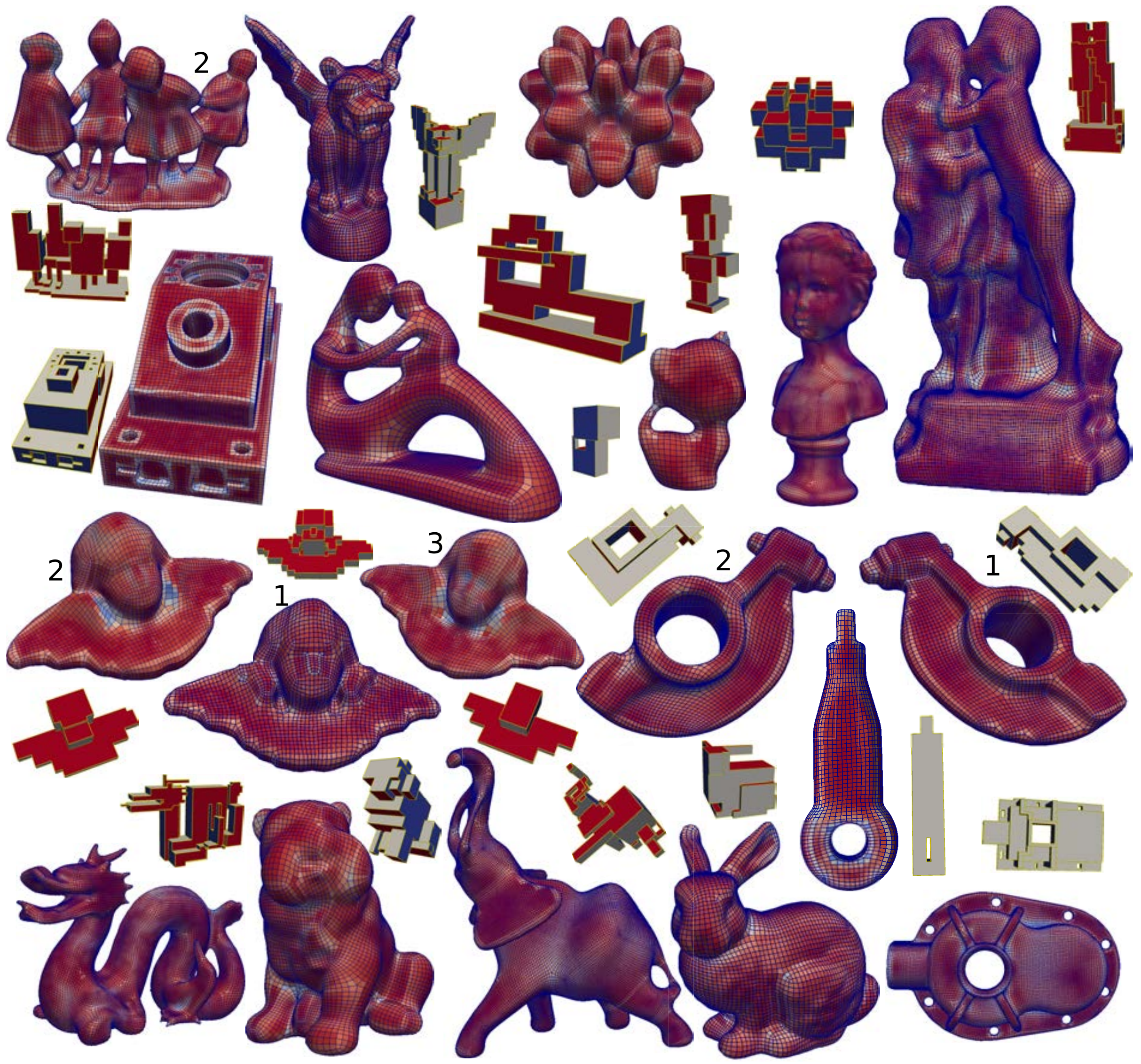


Fig. 17. Gallery of our polycube maps and hexahedral remeshing results. We colored the hexahedral elements according to their scaled Jacobian to help visual inspection of the results. (Model "gargoyle" ©VCG-ISTI, "rod" ©INRIA.)

ASPERT, N., SANTA-CRUZ, D., AND EBRAHIMI, T. 2002. Mesh: measuring errors between surfaces using the hausdorff distance. In *Multimedia and Expo, 2002. ICME '02. Proceedings. 2002 IEEE International Conference on*. Vol. 1. 705–708 vol.1.

AVRON, H., SHARF, A., GREIF, C., AND COHEN-OR, D. 2010. ℓ_1-sparse reconstruction of sharp point set surfaces. *ACM Transactions on Graphics 29,* 5, 135.

BENZI, M., GOLUB, G., AND LIESEN, J. 2005. Numerical solution of saddle point problems. *Acta numerica 14,* 1, 1–137.

BOTSCH, M., PAULY, M., GROSS, M., AND KOBBELT, L. 2006. Primo: coupled prisms for intuitive surface modeling. In *Symposium on Geometry Processing*. Vol. 256. 11–20.

CAMPEN, M., BOMMES, D., AND KOBBELT, L. 2012. Dual loops meshing: quality quad layouts on manifolds. *ACM Transactions on Graphics 31,* 4, 110.

CHEN, Y., DAVIS, T. A., HAGER, W. W., AND RAJAMANICKAM, S. 2008. Algorithm 887: Cholmod, supernodal sparse cholesky factorization and update/downdate. *ACM Trans. Math. Softw. 35,* 3 (Oct.), 22:1–22:14.

EL-ATTAR, R., VIDYASAGAR, M., AND DUTTA, S. 1979. An algorithm for ℓ_1-norm minimization with application to nonlinear ℓ_1–approximation. *SIAM Journal on Numerical Analysis 16,* 1, 70–86.

GARCIA, I., XIA, J., HE, Y., XIN, S.-Q., AND PATOW, G. 2013. Interactive applications for sketch-based editable polycube map. *IEEE Transactions on Visualization and Computer Graphics 19,* 7 (July), 1158–1171.

GOULD, N. I. M., ORBAN, D., AND TOINT, P. L. 2003. *An interior-point ℓ_1-penalty method for nonlinear optimization*. Technical Report RAL-TR-2003-022,Rutherford Appleton Laboratory.

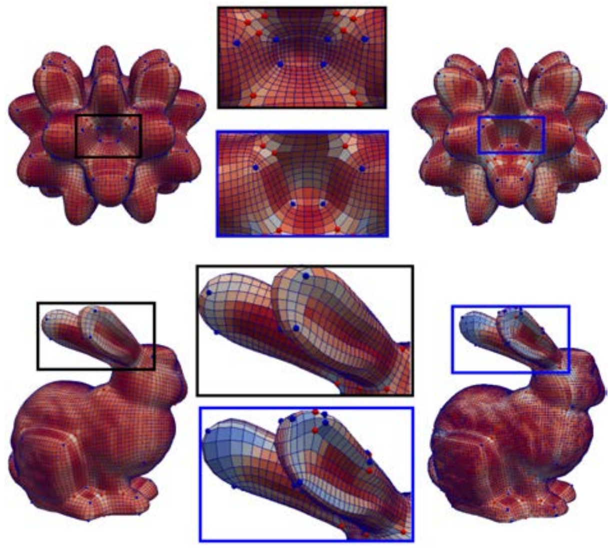


Fig. 15. Even for organic shapes, our method (left) generates better polycube structures than [Gregson et al. 2011] (right). Our orientation independent approach provides more symmetrical distribution of the polycube corners for the bumpy torus model; our distortion-driven global structure generation leads to better hex quality for the bunny model. (Models “bumpy torus, bunny” ©Max Planck Institute for Computer Science.)

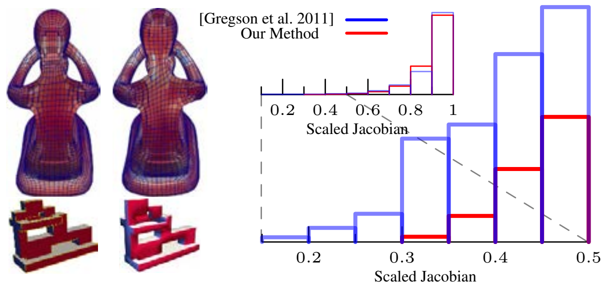


Fig. 16. Histogram of the scaled Jacobian on fertility model. The bottom histogram is a zoomed-in view of the top one in the range [0, 0.5]. Our result (left) contains 89.7% elements with scaled Jacobian > 0.8, and 0.251% elements with scale Jacobian < 0.5, while the result of [Gregson et al. 2011] (right) contains 87.4% and 0.744% elements under the same threshold respectively. (Model “fertility” ©Utrecht University.)

GREGSON, J., SHEFFER, A., AND ZHANG, E. 2011. All-hex mesh generation via volumetric polycube deformation. *Computer Graphics Forum 30,* 5, 1407–1416.

HE, Y., WANG, H., FU, C.-W., AND QIN, H. 2009. A divide-and-conquer approach for automatic polycube map construction. *Computers & Graphics 33*, 369–380.

HUANG, J., TONG, Y., WEI, H., AND BAO, H. 2011. Boundary aligned smooth 3d cross-frame field. *ACM Transactions on Graphics 30,* 6, 143.

HUBER, P. J. 1981. *Robust Statistics*. John Wiley & Sons, Inc.

IRVING, G., TERAN, J., AND FEDKIW, R. 2004. Invertible finite elements for robust simulation of large deformation. In *Symposium on Computer Animation*. 131–140.

KNÖPPEL, F., CRANE, K., PINKALL, U., AND SCHRÖDER, P. 2013. Globally optimal direction fields. *ACM Trans. Graph. 32,* 4 (July), 59:1–59:10.

LI, Y., LIU, Y., XU, W., WANG, W., AND GUO, B. 2012. All-hex meshing using singularity-restricted field. *ACM Trans. Graph. 31,* 6 (Nov.), 177:1–177:11.

LIN, J., JIN, X., FAN, Z., AND WANG, C. C. L. 2008. Automatic polycube-maps. In *Proceedings of the 5th international conference on Advances in geometric modeling and processing*. GMP’08. Springer-Verlag, Berlin, Heidelberg, 3–16.

LUKSAN, L., MATONOHA, C., AND VLCEK, J. 2007. Trust-region interior-point method for large sparse ℓ_1 optimization. *Optimization Methods Software 22,* 5 (Oct.), 737–753.

MYLES, A., PIETRONI, N., KOVACS, D., AND ZORIN, D. 2010. Feature-aligned T-meshes. *ACM Transactions on Graphics 29,* 4, 117.

PINAR, M. C. AND HARTMANN, W. M. 2006. Huber approximation for the non-linear ℓ_1 problem. *European Journal of Operational Research 169,* 3 (March), 1096–1107.

SANDIA, NATIONAL, L. 1997-2010. Mesh verification with verdict. `https://cubit.sandia.gov/public/verdict.html`.

SCHITTKOWSKI, K. 2008. NLPL1: A Fortran Implementation of an SQP Algorithm for Minimizing Sums of Absolute Function Values - User’s Guide.

SCHÖBERL, J. 1997. Netgen: An advancing front 2d/3d-mesh generator based on abstract rules. *Computing and visualization in science 1,* 1, 41–52.

TARINI, M., HORMANN, K., CIGNONI, P., AND MONTANI, C. 2004. Polycube-maps. *ACM Transactions on Graphics 23,* 3, 853–860.

TARINI, M., PUPPO, E., PANOZZO, D., PIETRONI, N., AND CIGNONI, P. 2011. Simple quad domains for field aligned mesh parametrization. *ACM Transactions on Graphics 30,* 6, 142.

WANG, H., JIN, M., HE, Y., GU, X., AND QIN, H. 2008. User-controllable polycube map for manifold spline construction. In *ACM Symposium on Solid and Physical Modeling*. 397–404.

YAO, C. AND LEE, T. 2008. Adaptive geometry image. *IEEE Transactions on Visualization and Computer Graphics 14,* 4, 948–960.