Cubic Stylization

HSUEH-TI DEREK LIU, University of Toronto, Canada
ALEC JACOBSON, University of Toronto, Canada

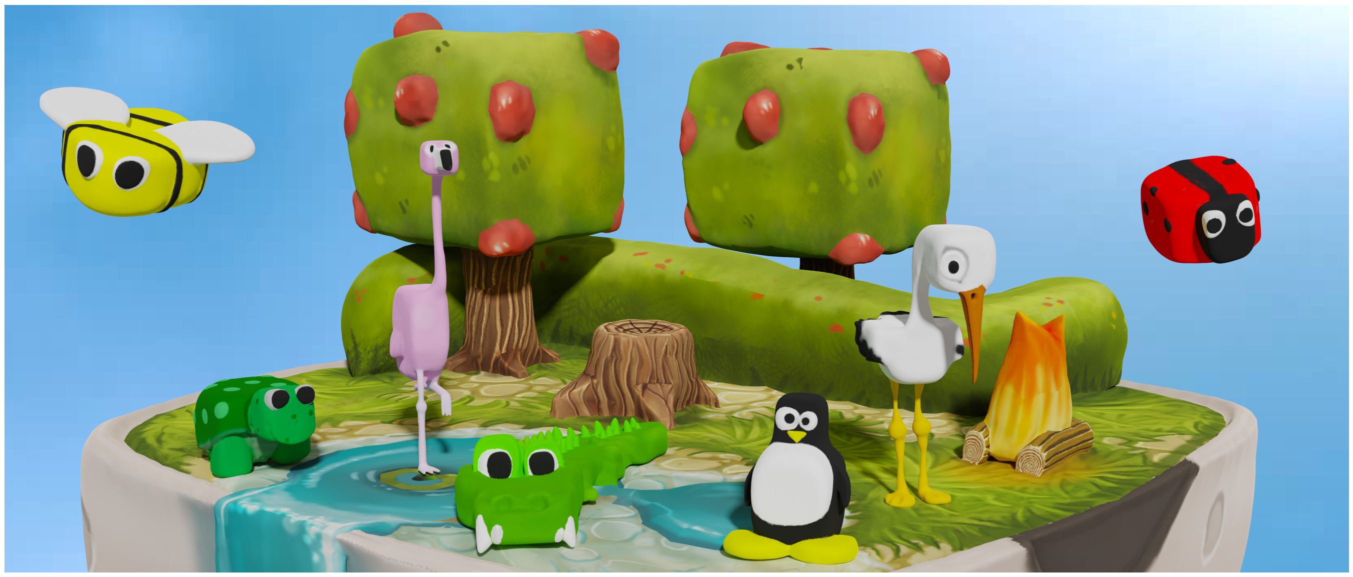


Fig. 1. *Cubic stylization* deforms a given 3D shape into the style of a cube while maintaining textures and geometric features. This can be used as a non-realistic modeling tool for creating stylized 3D virtual world. We obtain 3D assets from `sketchfab.com` by smeerws and Jesús Orgaz under CC BY 4.0.

We present a 3D stylization algorithm that can turn an input shape into the style of a cube while maintaining the *content* of the original shape. The key insight is that cubic style sculptures can be captured by the *as-rigid-as-possible* energy with an ℓ^1-regularization on rotated surface normals. Minimizing this energy naturally leads to a detail-preserving, cubic geometry. Our optimization can be solved efficiently without any mesh surgery. Our method serves as a non-realistic modeling tool where one can incorporate many artistic controls to create stylized geometries.

CCS Concepts: • **Computing methodologies** → **Mesh models**; **Mesh geometry models**.

Additional Key Words and Phrases: geometry processing, geometric stylization, shape modeling

ACM Reference Format:
Hsueh-Ti Derek Liu and Alec Jacobson. 2019. Cubic Stylization. *ACM Trans. Graph.* 38, 6, Article 197 (November 2019), 10 pages. https://doi.org/10.1145/3355089.3356495

Authors' addresses: Hsueh-Ti Derek Liu, University of Toronto, 40 St. George Street, Toronto, ON, M5S 2E4, Canada, hsuehtil@cs.toronto.edu; Alec Jacobson, University of Toronto, 40 St. George Street, Toronto, ON, M5S 2E4, Canada, jacobson@cs.toronto.edu.

1 INTRODUCTION

The availability of image stylization filters and *non-photorealistic rendering* techniques has dramatically lowered the barrier of creating artistic imagery to the point that even a non-professional user can easily create stylized images. In stark contrast, direct stylization of 3D shapes or *non-realistic modeling* has received far less attention. In professional industries such as visual effects and video games, trained modelers are still required to meticulously create non-realistic geometric assets. This is because investigating geometric styles is more challenging due to arbitrary topologies, curved metrics, and non-uniform discretization. The scarcity of tools to generate artistic geometry remains a major roadblock to the development of geometric stylization.

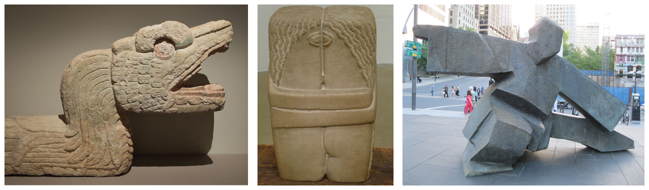


Fig. 2. The cubic style have been attracting artists' attention over centuries, such as the *Serpend à' Plumes* found in Chichén Itzá (left), *The Kiss* by Constantin Brâncuși (middle), and the *Taichi* by Ju Ming (right). We obtain images from `wikimedia.com` photographed by Jebulon under CC0 1.0, from `flickr.com` by Art Poskanzer under CC BY 2.0, and from `wikimedia.com` by Jeangagnon under CC BY-SA 3.0.

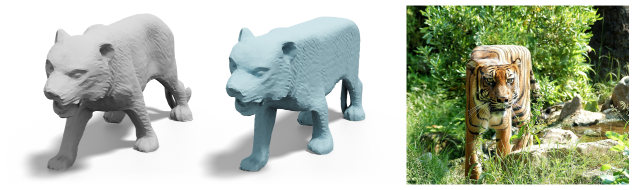


Fig. 3. A digital art – Anicube – by Aditya Aryanto produces cubic style images (right). Our method takes an input tiger (left) and outputs a "3D anicube" tiger while maintaining geometric details (middle). ©Aditya Aryanto (right). Used under permission.

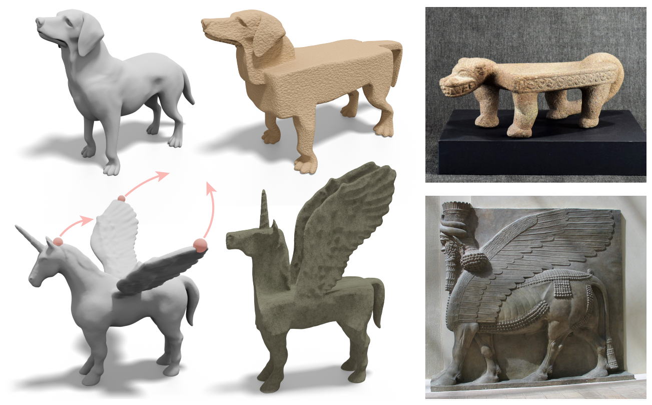


Fig. 4. One can control the cubic stylization by incorporating constraints. For instance, we can fix some parts of a shape to mimic the style of a Jaguar metate from ancient Costa Rica (top) or add point constraints to mimic the Assyrian Lamassu wall sculpture (bottom). ©Antiques & Artifacts LLC (top). Used under permission.

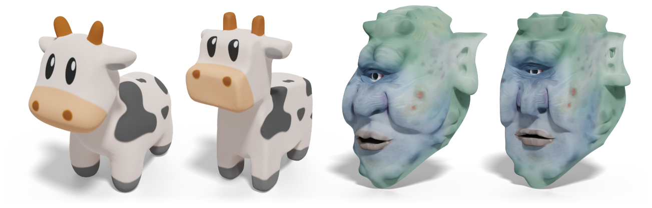


Fig. 5. Our cubic stylization requires no remeshing, thus vertex attributes such as textures are preserved during the optimization. Our ARAP term encourges locally isometric deformations to help maintain nice textures.

In this paper, we focus on the specific style of *cubic* sculptures. The cubic style is prevalent across art history, for instance the ancient sculptures from the post-classic era (900-1250 CE), Maya sculptures, block statues in Egypt, and modern abstract sculptures such as the ones from Constantin Brâncuși and Ju Ming (Fig. 2). In addition, the cubic style is a popular digital art, such as the award-winning *Anicube* by Aditya Aryanto (Fig. 3). Complementing their presence in art, cubic shapes also present themselves in fabrication and furniture purposes (Fig. 4). We contribute to the rich history of cubic sculpting by providing a stylization tool that takes a 3D shape as input and outputs a deformed shape that has the same style as cubic sculptures.

We present *cubic stylization* which formulates the task as an energy optimization that naturally preserves geometric details while cubifying a shape. Our proposed energy combines an *as-rigid-as-possible* (ARAP) energy with an ℓ^1 regularization. This energy can be minimized efficiently using the local-global approach with *alternating direction method of multipliers* (ADMM). This variational approach affords the flexibility of incorporating many artistic controls, such as applying constraints, non-uniform cubeness, and different global/local cube orientations (Sec. 4). Moreover, our method requires no remeshing (Fig. 5) and generalizes to polyhedral stylization (Fig. 24). Our proposed tool for non-realistic modeling goes beyond the 2D stylization and opens up the possibility of, for instance, creating non-realistic 3D worlds in virtual reality (Fig. 1).

2 RELATED WORK

Our work shares similar motivations to a large body of work on image stylization [Kyprianidis et al. 2013], non-photorealistic rendering [Gooch and Gooch 2001], and motion stylization [Hertzmann et al. 2009]. While their outputs are images or stylized animations, we take a 3D shape as input and output a stylized shape. Thus we focus our discussion on methods for processing geometry, including the study of geometric styles and deformation methods that share technical similarities.

Discriminative Geometric Styles. The growing interest in understanding geometric styles has been inspiring recent works on building *discriminative* models for style analysis. One of the main challenges is to define a similarity metric aligned with human perception. Many works propose to compare projected feature curves [Li et al. 2013; Yu et al. 2018], sub-components of a shape [Hu et al. 2017; Lun et al. 2015; Xu et al. 2010], or using learned features [Lim et al. 2016]. These models enable users to synthesize style compatible scenes [Liu et al. 2015] or transfer style components across shapes [Berkiten et al. 2017; Lun et al. 2016; Ma et al. 2014]. However, these methods are designed for discerning and transfering styles, instead of generating 3D stylized shapes directly.

Generative Geometric Styles. Direct 3D stylization has been an important topic in computer graphics. Many *generative* models have been proposed for producing specific styles, without relying on identifying and transferring style components from other shapes. This includes creating the collage art [Gal et al. 2007; Theobalt et al. 2007], voxel/lego art [Luo et al. 2015; Testuz et al. 2013], *neuronal homunculus* [Reinert et al. 2012], the manga style shapes [Shen et al. 2012], shape abstraction [Kratt et al. 2014; Mehra et al. 2009; Yumer and Kara 2012], and *bas-relief* sculptures [Bian and Hu 2011; Kerber et al. 2009; Schüller et al. 2014; Song et al. 2007; Weyrich et al. 2007]. While not pitched as stylization techniques, many geometric flows and filters can also be used for creating stylized geometry, such as creating edge-preserving smoothing geometry [Zhang et al. 2018], piece-wise planar [He and Schaefer 2013; Stein et al. 2018b] or developable shapes [Stein et al. 2018a], and stylized shapes prescribed by image filters [Liu et al. 2018] (see Fig. 6). Our method contributes to the field of direct 3D stylization, focusing on the style of cubic sculptures (Fig. 7).

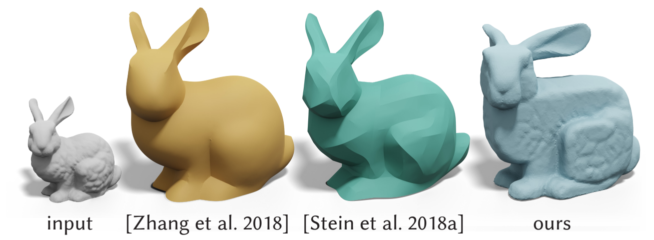


Fig. 6. Our energy-based deformation shares similarities with many energy-based geometric flows and mesh filters, such as the methods of [Zhang et al. 2018] and [Stein et al. 2018a].

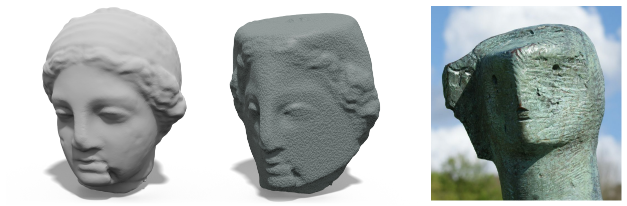


Fig. 7. Cubic style sculptures are common throughout history, such as the *Draped Seated Woman* by Henry Moore (right). Our cubic stylization offers an instrument to create cubic geometry (middle). We obtain the the photo from `flickr.com` photographed by puffin11k under CC BY-SA 2.0.

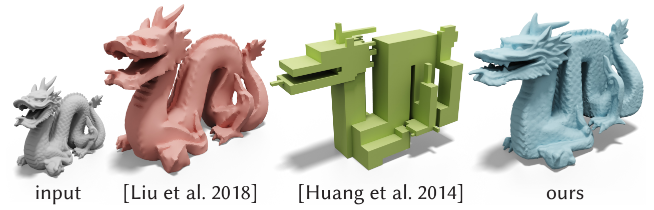


Fig. 8. Paparazzi [Liu et al. 2018] with image quantization and polycube method (e.g., [Huang et al. 2014]) can create cubic style shapes (red, green), but unlike our method (blue) they do not preserve geometric details.

Shape Deformation. Many works deal with the question of how to deform shapes given modeling constraints. One of the most popular choices is the ARAP energy [Chao et al. 2010; Igarashi et al. 2005; Liu et al. 2008; Sorkine and Alexa 2007], which measures local rigidity of the surface and leads to detail-preserving deformations. Not just deformations, similar formulations to ARAP can also be extended to other tasks such as constrained shape optimization [Bouaziz et al. 2012], parameterization [Liu et al. 2008], and simulating mass-spring systems [Liu et al. 2013]. Ever since, optimizing the ARAP energy has been substantially accelerated by a large amount of work, such as [Kovalsky et al. 2016; Peng et al. 2018; Rabinovich et al. 2017; Shtengel et al. 2017; Zhu et al. 2018]. However, having nearly interactive performance on highly detailed meshes still remains a major challenge. An alternative strategy to speed it up is to use the hierarchical deformation which optimizes ARAP on a low resolution model and then recover the original details back afterwards [Manson and Schaefer 2011]. This class of accelerations shares similar characteristics to multiresolution modeling (see [Garland 1999; Zorin 2006]). We take advantage of the ARAP energy for detail preservation and adapt the method of Manson and Schaefer [2011] to accelerate our cubic stylization to meshes with millions of faces.

Axis-Alignment in Polycube Maps. Axis-alignment is an important property for many geometry processing tasks, such as [Muntoni et al. 2018; Stein et al. 2019]. Especially, this concept is one of the main instruments in the construction of polycube maps [Tarini et al. 2004], including defining polycube segmentations [Fu et al. 2016; Livesu et al. 2013; Zhao et al. 2018] and the cost function for polycube deformations [Gregson et al. 2011; Huang et al. 2014]. Although polycube methods can obtain cubic geometry, they fail to preserve detail (Fig. 8) because they are not desirable for intended applications such as parameterization and hexahedral meshing [Cherchi et al. 2016; Fang et al. 2016; García Fernández et al. 2013; He et al. 2009; Lin et al. 2008; Wang et al. 2007, 2008; Yu et al. 2014].

One tempting direction of creating cubic geometry is to use voxelization. However, voxelization fails to capture the details depicted by the artists and cannot capture the wide spectrum of cubeness across cubic sculptures. Another tempting direction is to recover geometric features from the polycube results. This would lead to a multi-step algorithm and suffer from limitations of particular detail encoding schemes (e.g., bump maps). Even if we stop the polycube algorithm earlier such as the method of [Gregson et al. 2011] to maintain details, it does not provide a satisfactory solution (see the inset for a comparison with Fig. 5 in [Gregson et al. 2011]). More importantly, many artistic controls in Sec. 4 would be nontrivial to add on. Building stylization on top of polycube methods would also suffer from slow performance. For instance, Huang et al. [2014] propose a polycube method that minimizes the ℓ^1-norm of the normals on the deformed tetrahedral mesh with ARAP for regularization. Their formulation involves minimizing a complicated non-linear function and requires minutes to hours to optimize. Thus a stylization built on top of this method would be even slower. In contrast, our formulation is a single energy optimization which can easily incorporate many artistic controls (Sec. 4). Our energy is similar to the polycube energy of [Huang et al. 2014] in that we also minimize the ARAP energy with a ℓ^1 regularization, but the key difference is that we define the ℓ^1-norm on the *rotated normals* of the *original* mesh instead. This allows us to optimize our energy much faster using the local-global approach with ADMM in only a few seconds (Table 1).

3 METHOD

The input of our method is a manifold triangle mesh with/without boundaries. Our method outputs a *cubified* shape where each subcomponent has the style of an axis-aligned cube. Meanwhile, our stylization will maintain the geometric details of the original mesh.

Let V be a $|\mathsf{V}| \times 3$ matrix of vertex positions at the rest state and $\widetilde{\mathsf{V}}$ be a $|\mathsf{V}| \times 3$ matrix containing the deformed vertex positions. We denote $\mathsf{d}_{ij} = [\mathsf{v}_j - \mathsf{v}_i]^\top$ and $\widetilde{\mathsf{d}}_{ij} = [\widetilde{\mathsf{v}}_j - \widetilde{\mathsf{v}}_i]^\top$ be the edge vectors between vertices i, j at the rest and deformed states respectively.

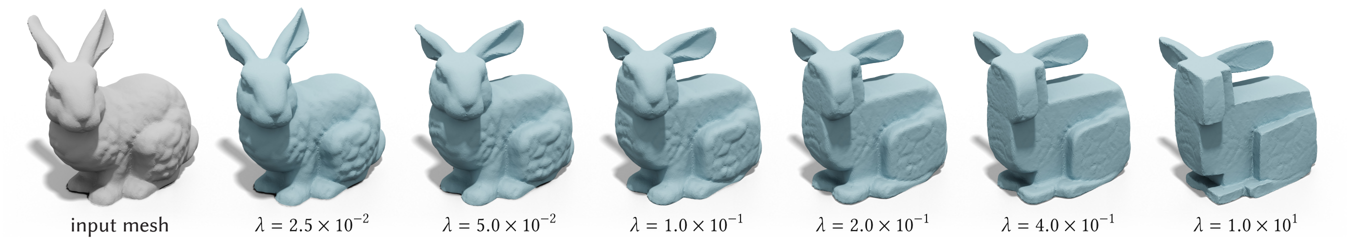
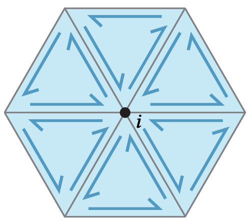

input mesh
$\lambda = 2.5 \times 10^{-2}$
$\lambda = 5.0 \times 10^{-2}$
$\lambda = 1.0 \times 10^{-1}$
$\lambda = 2.0 \times 10^{-1}$
$\lambda = 4.0 \times 10^{-1}$
$\lambda = 1.0 \times 10^{1}$

Fig. 9. One can control the CUBENESS by changing the λ parameter in Eq. 1.

The energy for our cubic stylization is as follows

$$\underset{\widetilde{V}, \{R_i\}}{\text{minimize}} \sum_{i \in V} \underbrace{\sum_{j \in \mathcal{N}(i)} \frac{w_{ij}}{2} \|R_i d_{ij} - \widetilde{d}_{ij}\|_F^2}_{\text{ARAP}} + \underbrace{\lambda a_i \|R_i \hat{n}_i\|_1}_{\text{CUBENESS}}. \tag{1}$$

The first term is the ARAP energy [Sorkine and Alexa 2007], where R_i is a 3-by-3 rotation matrix, w_{ij} is the cotangent weight [Pinkall and Polthier 1993], and $\mathcal{N}(i)$ denotes the "spokes and rims" edges of the ith vertex [Chao et al. 2010] (see the inset). In the second term, $\hat{n}_i$ denotes the unit area-weighted normal vector of a vertex i in $\mathbb{R}^3$. The $a_i \in \mathbb{R}^+$ is the barycentric area of vertex i, which is crucial for λ to exhibit the similar cubeness across different mesh resolutions. Intuitively, minimizing the ℓ^1-norm of the rotated normal encourages $R_i \hat{n}_i$ to align with one of coordinate axes because ℓ^1-norm encourages sparsity. Combining the two, the optimal rotation $\{R_i^\star\}$ would simultaneously preserve the local structure (ARAP) and encourage axis alignment (CUBENESS).



We adapt the standard local-global update strategy to optimize our energy [Sorkine and Alexa 2007] (see Alg. 1). Our global step, updating $\widetilde{V}$, is achieved by solving a linear system, the same as the Equation 9 in Sorkine and Alexa [2007]. Our local step, finding the optimal rotation, is however different from the previous literature due to the ℓ^1 term.

3.1 Local-Step

Our local step for each vertex i can be written as

$$R_i^\star = \underset{R_i \in SO(3)}{\arg\min} \ \frac{1}{2} \|R_i D_i - \widetilde{D}_i\|_{W_i}^2 + \lambda a_i \|R_i \hat{n}_i\|_1, \tag{2}$$

where W_i is a $|\mathcal{N}(i)| \times |\mathcal{N}(i)|$ diagonal matrix of cotangent weights, D_i and $\widetilde{D}_i$ are $3 \times |\mathcal{N}(i)|$ matrices of rim/spoke edge vectors at the rest and deformed states respectively. We denote $\|X\|_Y^2 = \mathrm{Tr}(XYX^\top)$ for notational convenience. By setting $z = R_i \hat{n}_i$, we can rewrite Eq. 2 as

$$\begin{aligned} \underset{z, R_i \in SO(3)}{\text{minimize}} \quad & \frac{1}{2} \|R_i D_i - \widetilde{D}_i\|_{W_i}^2 + \lambda a_i \|z\|_1 \\ \text{subject to} \quad & z - R_i \hat{n}_i = 0. \end{aligned} \tag{3}$$

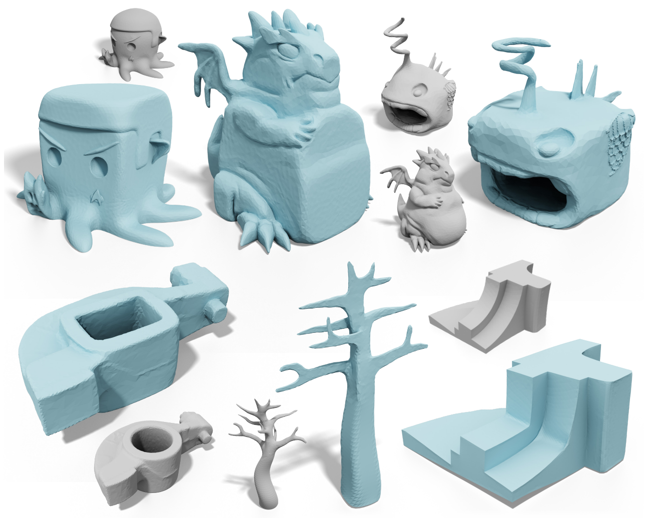


Fig. 10. We turn 3D shapes into the cubic style (blue) with Alg. 1. ©Angelo Tartanian (top left), Splotchy Ink (top), Dan Slack (top right) under CC BY.

Eq. 3 is a standard ADMM formulation. We solve this local step using the scaled-form ADMM updates [Boyd et al. 2011]:

$$R_i^{k+1} \leftarrow \underset{R_i \in SO(3)}{\arg\min} \ \frac{1}{2} \|R_i D_i - \widetilde{D}_i\|_{W_i}^2 + \frac{\rho^k}{2} \|R_i \hat{n}_i - z^k + u^k\|_2^2 \tag{4}$$

$$z^{k+1} \leftarrow \underset{z}{\arg\min} \ \lambda a_i \|z\|_1 + \frac{\rho^k}{2} \|R_i^{k+1} \hat{n}_i - z + u^k\|_2^2 \tag{5}$$

$$\tilde{u}^{k+1} \leftarrow u^k + R_i^{k+1} \hat{n}_i - z^{k+1} \tag{6}$$

$$\rho^{k+1}, u^{k+1} \leftarrow \textit{update}\,(\rho^k) \tag{7}$$

where $\rho \in \mathbb{R}_+$ is the penalty and u is the scaled dual variable.

Eq. 4 is an instance of the *orthogonal Procrustes* [Gower et al. 2004]

$$R_i^{k+1} \leftarrow \underset{R_i \in SO(3)}{\arg\max} \ \mathrm{Tr}(R_i M_i)$$

$$M_i = \begin{bmatrix} D_i & \hat{n}_i \end{bmatrix} \begin{bmatrix} W_i & \\ & \rho^k \end{bmatrix} \begin{bmatrix} \widetilde{D}_i^\top \\ (z^k - u^k)^\top \end{bmatrix}.$$

One can derive the optimal R_i from the singular value decomposition of $M_i = \mathcal{U}_i \Sigma_i \mathcal{V}_i^\top$:

$$R_i^{k+1} \leftarrow \mathcal{V}_i \mathcal{U}_i^\top, \tag{8}$$

up to changing the sign of the column of $\mathcal{U}_i$ so that $\det(R_i) > 0$.

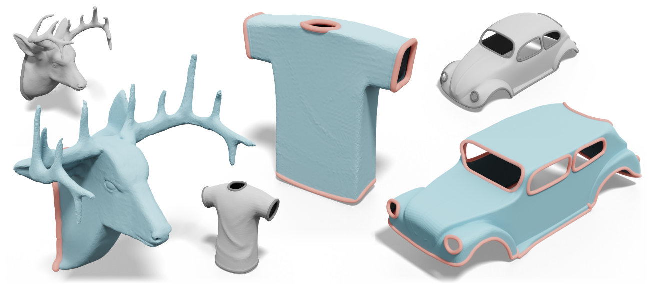


Fig. 11. We can also turn meshes with boundaries (red) into the cubic style. ©Takeshi Murata (left) under CC BY.

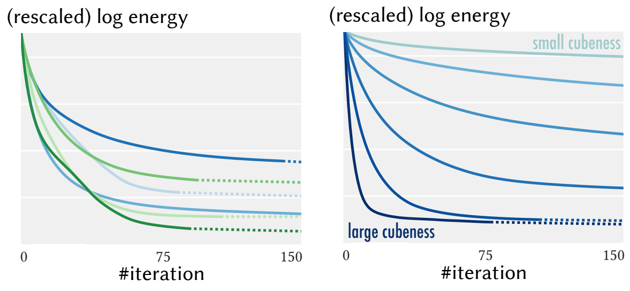


Fig. 12. We show the convergence behavior of different meshes in Fig. 10 (left, blue), Fig. 16 (left, green), and different cubenesses in Fig. 9 (right). Note that the dotted line imply the optimization has stopped.

Eq. 5 is an instance of the *lasso* problem [Boyd et al. 2011; Tibshirani 1996], which can be solved with a *shrinkage* step:

$$\mathrm{z}^{k+1} \leftarrow \mathcal{S}_{\lambda a_i/\rho^k}(\mathrm{R}_i^{k+1}\hat{\mathrm{n}}_i + \mathrm{u}^k) \tag{9}$$
$$\mathcal{S}_\kappa(\mathrm{x})_j = (1 - \kappa/|x_j|)_+ \, x_j$$

We update the penalty ρ (Eq. 7) according to Sec. 3.4.1 in [Boyd et al. 2011] where u needs to be rescaled accordingly after updating ρ.

In short, local fitting is performed by running Eq. 8, 9, 6, and 7 iteratively until the norm of primal/dual residuals are small. Warm starting the local-step parameters from the previous iteration can significantly speed up the optimization. Specifically, we initialize z, u with zeros, and set the initial $\rho = 10^{-4}$, $\epsilon^{\text{abs}} = 10^{-5}$, $\epsilon^{\text{rel}} = 10^{-3}$, $\mu = 10$, and $\tau^{\text{incr}} = \tau^{\text{decr}} = 2$ (the same notation as used in Sec. 3 of [Boyd et al. 2011]). Then z, u, ρ are reused in consecutive iterations. Note that for extremely large λ one may need to increase the initial value of ϵ^{abs} accordingly in order to avoid bad local minima. We stop the optimization when the relative displacement, the infinity norm of relative per vertex displacements, is lower than 3×10^{-3} (see Fig. 12 for the convergence plots). More elaborate stopping criteria, such as the method of [Zhu et al. 2018], could also be used.

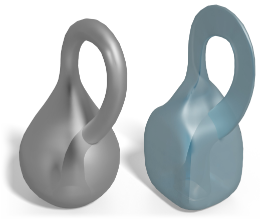


Fig. 14. Our method can cubify non-orientable surfaces such as the Klein bottle.

At this point we have completed the cubic stylization algorithm summarized in Alg. 1, enabling us to efficiently create cubified shapes (see Fig. 10). In Fig. 11 and 14 we show that this formulation is applicable to meshes with boundaries and non-orientable surface respectively. As the CUBENESS is dependent to the orientation of the mesh, one can apply different rotations to control how the stylization runs (Fig. 13). We expose the weighting λ to be a design parameter controlling the cubeness of a shape (Fig. 9).

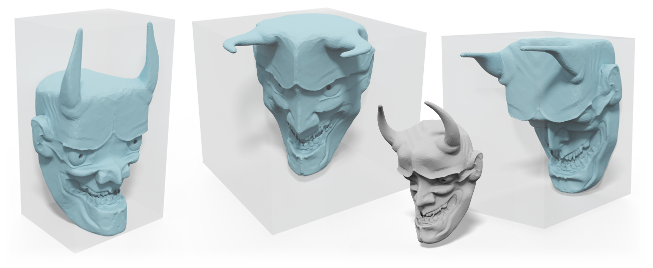


Fig. 13. The global orientation of the shape influences the ℓ^1 term in Eq. 1. Applying different rotations to the mesh lead to different results. ©My Dog Justice under CC BY.

Algorithm 1: *Cube Stylization* (λ)

```
Input  : A triangle mesh V, F
Output: Deformed vertex positions $\widetilde{V}$
1. $\widetilde{V} \leftarrow V$
2. while not converge do
3.     $R \leftarrow$ local-step$(V, \widetilde{V}, \lambda)$
4.     $\widetilde{V} \leftarrow$ global-step$(R)$
```

Algorithm 2: *Fast Cube Stylization* (λ, m)

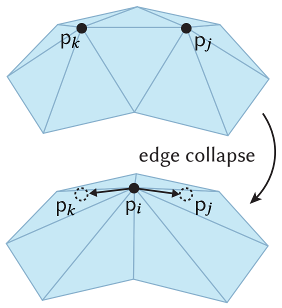
```
Input  : A triangle mesh V, F
Output: Deformed vertex positions $\widetilde{V}$
// pre-processing
1. $m \leftarrow$ target number of faces
2. $V_c, F_c \leftarrow$ edge collapses$(V, F, m)$
// cubic stylization
3. $\widetilde{V}_c \leftarrow V_c$
4. while not converge do
5.     $R \leftarrow$ local-step$(V_c, \widetilde{V}_c, \lambda)$
6.     $\widetilde{V}_c \leftarrow$ global-step$(R)$
7. $\widetilde{V}, F \leftarrow$ affine vertex splits$(\widetilde{V}_c, F_c)$
```

However, the "vanilla" cube stylization shares the same caveat as other distortion minimization algorithms: having slow runtime on high resolution meshes.

3.2 Affine Progressive Meshes

Manson and Schaefer [2011] propose a hierarchical approach to accelerate ARAP deformations. The main idea is to deform a low-resolution model and recover the details back after convergence.

Specifically, Manson and Schaefer [2011] propose a progressive mesh [Hoppe 1996] representation which first simplifies a given mesh via a sequence of edge collapses, and then represents the mesh as its coarsest form together with a sequence of vertex splits. After applying some deformations to the coarsest mesh, each "deformed" vertex split is computed by fitting the best local rigid transformation. This approach is suitable for deformations that are locally rigid (e.g., ARAP), but our cubic stylization is *less* rigid for larger λ.



So we fit the best *affine* transformation in each vertex split, rather than rigid transformations. Specifically, in each edge collapse we store the displacement vectors from the newly inserted vertex p_i to the endpoints p_j, p_k (see the inset) together with a matrix A:

$$A = (Q_i Q_i^\top)^{-1} Q_i.$$

Q_i is a $3 \times |\mathcal{N}(i)|$ matrix where each column is the vector from p_i to one of its one-rings neighbors $\mathcal{N}(i)$. If $(Q_i Q_i^\top)$ is singular (e.g., in planar regions), we remedy the issue with the Tikhonov regularization [Tikhonov et al. 2013]. Then A is used to computed the deformed displacements for each vertex split as

$$\widetilde{p}_j - \widetilde{p}_i = \widetilde{Q}_i A^\top (p_j - p_i),$$

where $\widetilde{p}_i$ denotes the position of vertex i in the cubified coarsened shape, and $\widetilde{Q}_i$ is a $3 \times |\mathcal{N}(i)|$ matrix containing vectors from $\widetilde{p}_i$ to its one-rings neighbors.

Affine progressive meshes allows us to losslessly recover the original meshes undergoing affine transformations. For smooth non-affine transformations such as our cube stylization, it could still be approximately recovered (see Fig. 15). We summarize our cubic stylization with the affine progressive mesh in Alg. 2. Note that the edge collapses is just a pre-processing step. In the online stage, one only needs to run cubic stylization on the coarsest mesh and then apply a sequence of vertex splits to visualize the result on the original resolution. This offers a huge speed-up when interacting the parameter λ on highly detailed models (see Fig. 16).

An interesting observation is that the number of faces m in the coarsest mesh not only controls the runtime, but implicitly controls the frequency level of geometric details that gets preserved. In Fig. 17 we show that, under the same λ, a smaller m keeps details across a wider frequency range; in contrast, a larger m only keeps details at higher frequencies. Therefore one can manipulate the level of preserved features by playing with m.

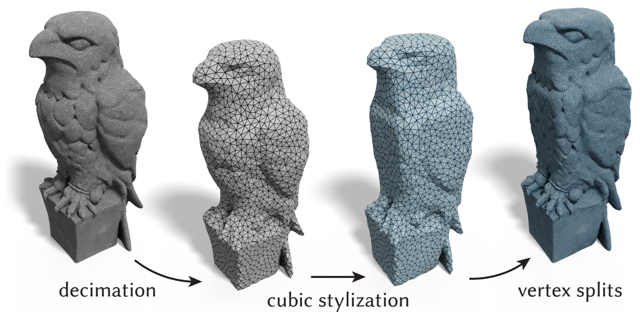


Fig. 15. Affine progressive meshes allow us run cubic stylization on a low-resolution model and then recover original details when converged. ©Colin Freeman under CC BY.

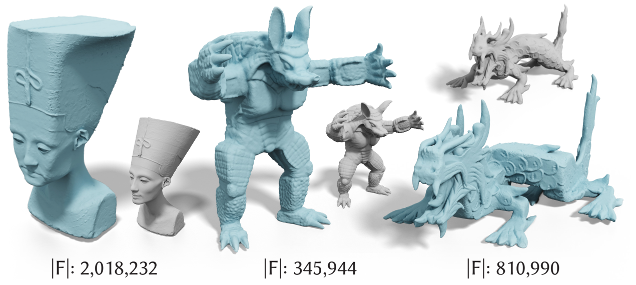


Fig. 16. With the affine progressive meshes, we can scale the cubic stylization to meshes with millions of faces. The Nefertiti mesh (left) was scanned by Nora Al-Badri and Jan Nikolai Nelles from the Nefertiti bust.

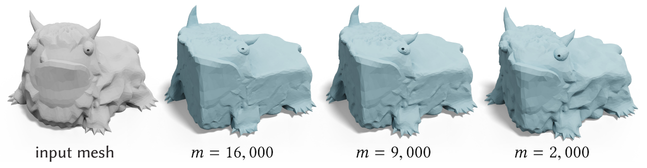
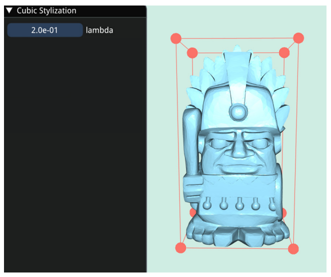


Fig. 17. The number of faces m used in the decimated mesh not only controls the runtime but also the frequency level of details that get preserved. ©Joseph Larson under CC BY.

3.3 Implementation

We implement the cubic stylization in C++ using LIBIGL [Jacobson et al. 2018] and evaluate our runtime on a MacBook Pro with an Intel i5 2.3GHz processor. Table 1 lists the parameters and the runtime of our stylization in Fig. 10 (top) and Fig. 16. We test our methods on meshes in the *Thingi10K* [Zhou and Jacobson 2016] and show that we can obtain stylized geometry within a few seconds. This is important for users to receive quick feedback on their parameter choices and iterate on their designs, such as the cubeness λ in Fig. 9 and the the level of details m in Fig. 17.



User study. We prototype a user interface (see the inset) to conduct an informal user study with six participants (4 male, 2 female) between the ages of 24 and 29. Participant 3D modeling experience ranged from none (complete novice) to three years of hobbyist use. Each participant was instructed for three minutes on how to use our software to load a mesh and control the cubeness parameter λ. Then we asked them to cubify a shape of their choosing from a collection of ten shapes. The results of their work is show in Fig. 18. All users reported that they were satisfied with the cubeness of their resulting shape. One user said

Table 1. For each example in Fig. 10 and Fig. 16, we report the number of faces in the original model ($|F|$), $l1$ weight (λ), number of faces of the coarsest mesh (m), number of iterations (*Iters.*), pre-processing time (*Pre.*), and runtime at the online stage (*Runtime*).

Model	$\|F\|$	λ	m	*Iters.*	*Pre.*	*Runtime*
Fig. 10, left	39K	0.20	n/a	106	n/a	**5.08s**
Fig. 10, mid.	41K	0.20	n/a	93	n/a	**4.50s**
Fig. 10, right	21K	0.4	n/a	86	n/a	**2.26s**
Fig. 16, left	2018K	0.20	20K	83	64.19*s*	**3.93s**
Fig. 16, mid.	346K	0.40	20K	222	10.69*s*	**4.59s**
Fig. 16, right	811K	0.30	40K	173	30.44*s*	**8.38s**

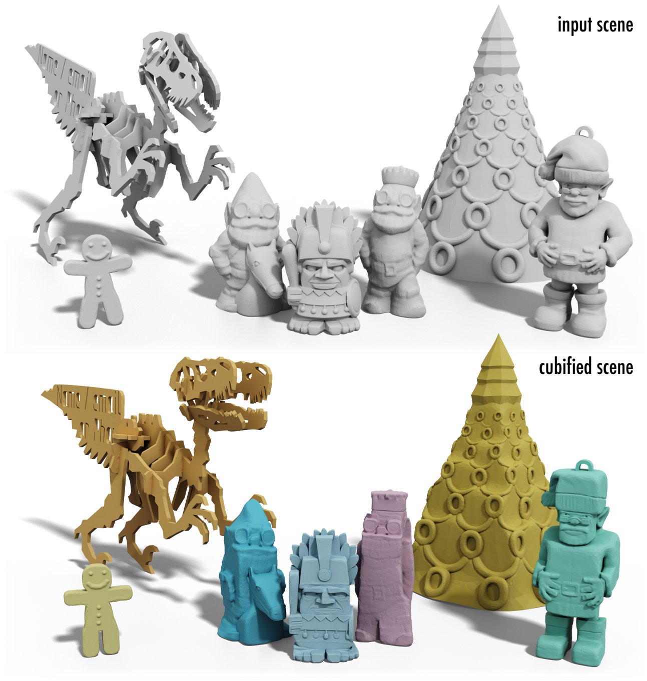


Fig. 18. Even non-professional users can effortlessly turn an input scene (top) into a cubified scene (bottom). Different colors are results created by different users. From left to right, ©Peter Leppik, Cleven, TerenceKing, MakerBot, TerenceKing, PerryEngel, and Christina Chun under CC BY.

that controlling the cubeness of their resulting shape is very easy because it only requires tuning a single parameter.

4 ARTISTIC CONTROLS

In addition to the two parameters λ, m, we expose many variants of our stylization to incorporate artistic controls. As a non-realistic modeling tool, this is important for users to realize their creativity.

We first focus our discussion on a variety of artistic controls that are related to the cubeness parameter λ. Although Eq. 1 only has a single λ for an entire shape, we can actually specify different λ_i for each vertex independently to have non-uniform cubeness, which leads to the expression $\lambda_i a_i \|\mathrm{R}_i \hat{\mathrm{n}}_i\|_1$. In Fig. 19, we use this approach to make the back of the sheep much more cubic than the rest of the shape to create an ottoman-like geometry. We can also specify the non-uniform cubeness λ_i in a different way, instead of painting on the surface directly. In Fig. 20 we *paint* a function on the Gauss map in which the surface normal pointing towards left has higher cubeness. When we map this function back to the surface, we can have a cubified owl that is more cubic when initial normals pointing towards the left and less cubic when pointing towards the right. Similarly, we can have different $\lambda_x, \lambda_y, \lambda_z$ for different axes. In Fig. 21, we replace the CUBENESS in Eq. 1 with $a_i(\lambda_x|(\mathrm{R}_i\hat{\mathrm{n}}_i)_x| + \lambda_y|(\mathrm{R}_i\hat{\mathrm{n}}_i)_y| + \lambda_z|(\mathrm{R}_i\hat{\mathrm{n}}_i)_z|)$ and specify different values for each $\lambda_x, \lambda_y, \lambda_z$ to have the style of a rectangular prism.

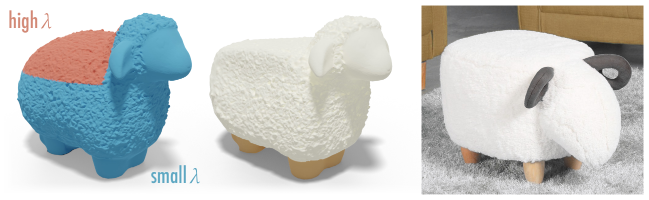


Fig. 19. We vary λ across the surface to have different cubeness for different parts. We apply higher λ on the red region and smaller λ for the blue region to create an ottoman-like shape (middle). ©pmoews under CC BY.

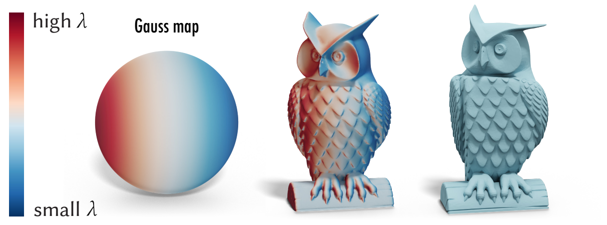


Fig. 20. We can paint the λ function on the Gauss map to have non-uniform λ over the surface. In the figure, we have higher λ for the original normals pointing towards left, and vice versa. ©Tom Cushwa under CC BY.

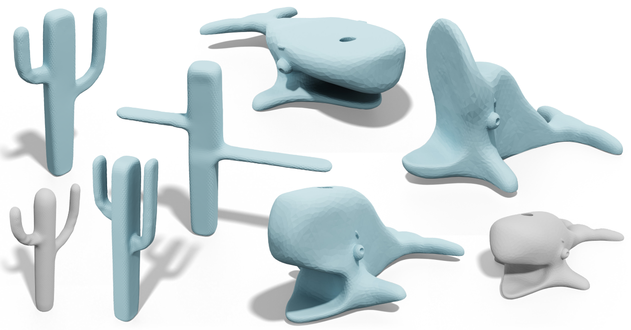


Fig. 21. We can vary λ for different axes to turn inputs into the biased cubic style (blue) towards x,y,z axes respectively. ©MakerBot (right) under CC BY.

If one wants to fix certain parts of the shape, we can easily add constraints in the global step, the same way as the method of Sorkine and Alexa [2007]. In Fig. 4 we add the parts constraint by fixing the position of some vertices when solving the linear system; we add the points constraint by specifying some deformed vertices $\widetilde{\mathrm{V}}_i$ at user-desired positions. We can also use the same methodology to

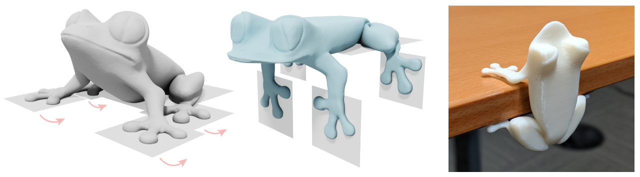


Fig. 22. We constrain certain parts of the geometry lying on certain planes to create a 3D printed table clinger (right). ©Morena Protti under CC BY.

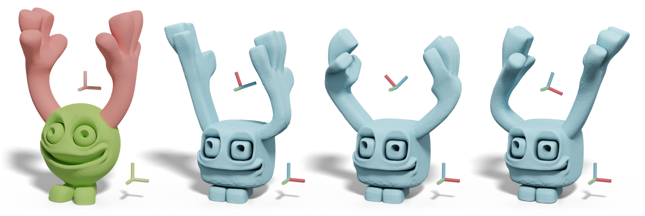


Fig. 23. We can define the ℓ^1-norm on different coordinate systems for different parts of the shape, instead of using the world coordinates. In the figure the hands and the body use different coordinate systems (left). By changing them, we can vary the cube orientations for different parts. ©David Hagemann under CC BY.

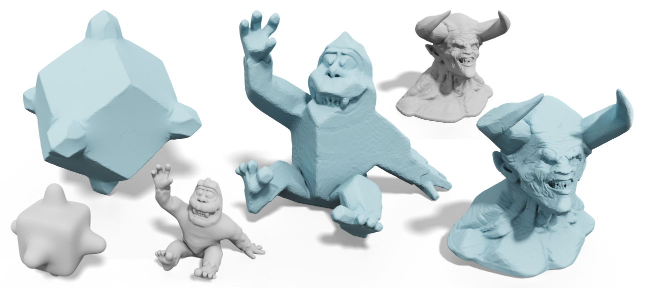


Fig. 24. We apply a coordinate transformation inside the ℓ^1-norm to generalize cubic stylization to polyhedrons. ©Proto Paradigm (middle), Ola Sundberg (right) under CC BY.

constrain some parts of the geometry lying on certain planes. For instance, setting $(\widetilde{\mathrm{V}}_i)_x = 0$ can force vertex i lying on the yz-plane. In Fig. 22 we use this plane constraint to create a table clinger.

In addition, one can utilize the property of the ℓ^1-norm to have different artistic effects. Because the CUBENESS term is orientation dependent, in Fig. 13 we can apply different rotations to the mesh before the stylization to control the results. Rather than rotating the mesh, another way is to encode the normal vector in a different coordinate system $\lambda a_i \|\mathrm{R}_i \hat{\mathrm{n}}_i^{\mathrm{local}}\|_1$, where we use $\hat{\mathrm{n}}_i^{\mathrm{local}}$ to denote the user-desired coordinate system for vertex i. This perspective allows us to define the ℓ^1-norm on different coordinate systems for different parts of the shape to obtain different cube orientations (Fig. 23). Beyond the cubic stylization, in Fig. 24, 25 we apply a coordinate transformation B inside the ℓ^1-norm $\lambda a_i \|\mathrm{B}\mathrm{R}_i \hat{\mathrm{n}}_i\|_1$ to achieve polyhedral stylization, for which we provide the details in App. A. Once we obtain the stylized shapes, they are ready to be used by standard deformation techniques in animations (Fig. 26).

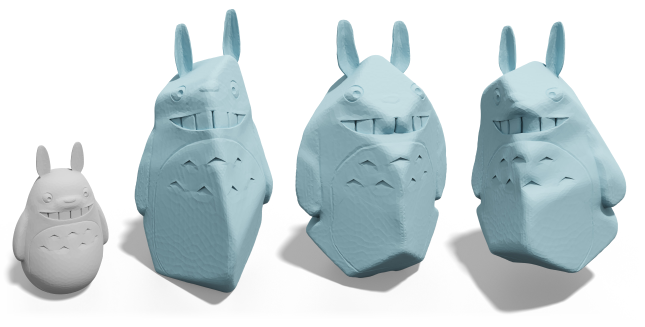


Fig. 25. We apply non-symmetric coordinate transformations inside the ℓ^1-norm to create irregular polyhedral stylization. ©Johannes under CC BY.

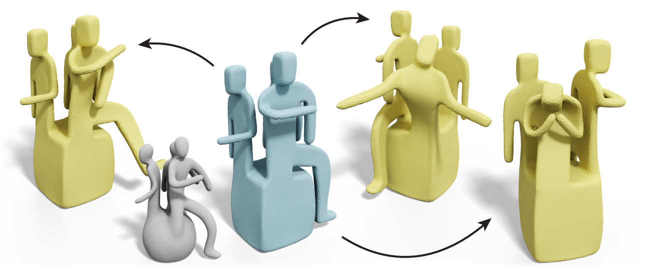


Fig. 26. Once we have the cubic geometry (blue), standard deformation techniques (e.g., [Sorkine and Alexa 2007]) can be used to manipulate the cubified shape (yellow).

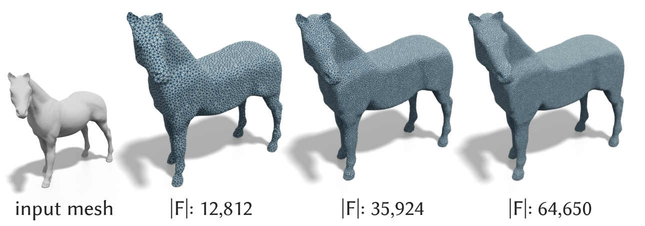


Fig. 27. Although exhibiting similar cubenesses, our stylization is still not invariant to different resolutions.

5 LIMITATIONS & FUTURE WORK

Accelerating the stylization to real-time would enable faster iterations between designs. Developing a more robust stylization to for bad quality triangles, non-manifold meshes, or even point cloud could be useful for stylizing real-world geometric data. Guaranteeing results to be self-intersection free would be desirable for downstream tasks. Extending our energy to be invariant to discretizations could achieve more consistent results across different resolutions (see Fig. 27). Extending to quadrilateral meshes and NURBS surfaces could benefit existing modeling or engineering design softwares. Generalizing to volumetric meshes could have a better volume preservation. Exploring different deformation energies and ℓ^p-norm could lead to novel stylization tools for non-realistic modeling. Beyond generating stylized shapes, the mathematical expression of the cubic geometry could offer insights toward understanding more intricate styles. For instance, *Cubism* has been considered as a revolutionized artistic style for paintings and sculptures. Cubism has appeared since the early 20th century. Since then, several attempts have tried to describe [Henderson 1983] and generate Cubist art [Corker-Marin et al. 2018; Wang et al. 2011], but more efforts still required to offer scientific explanations to a wide variety of Cubist art. Our cubic stylization only focuses on a specific style. We hope this could inspire future attempts to capture different sculpting styles such as those presented in African art, or even a generic approach to create different styles in an unified framework.

ACKNOWLEDGMENTS

Our research is funded in part by New Frontiers of Research Fund (NFRFE–201), the Ontario Early Research Award program, NSERC Discovery (RGPIN2017–05235, RGPAS–2017–507938), the Canada Research Chairs Program, the Fields Centre for Quantitative Analysis and Modelling and gifts by Adobe Systems, Autodesk and MESH Inc. We thank members of Dynamic Graphics Project at the University of Toronto; Michael Tao and Wen-Hsiang Tsai for project motivations; David I.W. Levin and Yotam Gingold for ideas on the artistic controls and the user study; Oded Stein for sharing results; Rahul Arora for fabricating stylized shapes; Leonardo Sacht and Silvia Sellán for proofreading; Omid Poursaeed, Rahul Arora, Whitney Chiu, Yang Zhou, Yifan Wang, Youssef Alami Mejjati, and Zhicong Lu for participating in the user study.

REFERENCES

Sema Berkiten, Maciej Halber, Justin Solomon, Chongyang Ma, Hao Li, and Szymon Rusinkiewicz. 2017. Learning detail transfer based on geometric features. In *Computer Graphics Forum*, Vol. 36. Wiley Online Library, 361–373.

Zhe Bian and Shi-Min Hu. 2011. Preserving detailed features in digital bas-relief making. *Computer Aided Geometric Design* 28, 4 (2011), 245–256.

Sofien Bouaziz, Mario Deuss, Yuliy Schwartzburg, Thibaut Weise, and Mark Pauly. 2012. Shape-up: Shaping discrete geometry with projections. In *Computer Graphics Forum*, Vol. 31. Wiley Online Library, 1657–1667.

Stephen Boyd, Neal Parikh, Eric Chu, Borja Peleato, Jonathan Eckstein, et al. 2011. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends® in Machine learning* 3, 1 (2011), 1–122.

Isaac Chao, Ulrich Pinkall, Patrick Sanan, and Peter Schröder. 2010. A simple geometric model for elastic deformations. *ACM transactions on graphics (TOG)* 29, 4 (2010), 38.

Gianmarco Cherchi, Marco Livesu, and Riccardo Scateni. 2016. Polycube simplification for coarse layouts of surfaces and volumes. In *Computer Graphics Forum*, Vol. 35. Wiley Online Library, 11–20.

Quentin Corker-Marin, Alexander Pasko, and Valery Adzhiev. 2018. 4D Cubism: Modeling, Animation, and Fabrication of Artistic Shapes. *IEEE computer graphics and applications* 38, 3 (2018), 131–139.

Xianzhong Fang, Weiwei Xu, Hujun Bao, and Jin Huang. 2016. All-hex meshing using closed-form induced polycube. *ACM Transactions on Graphics (TOG)* 35, 4 (2016), 124.

Xiao-Ming Fu, Chong-Yang Bai, and Yang Liu. 2016. Efficient volumetric polycube-map construction. In *Computer Graphics Forum*, Vol. 35. Wiley Online Library, 97–106.

Ran Gal, Olga Sorkine, Tiberiu Popa, Alla Sheffer, and Daniel Cohen-Or. 2007. 3D collage: expressive non-realistic modeling. In *Proceedings of the 5th international symposium on Non-photorealistic animation and rendering*. ACM, 7–14.

Ismael García Fernández, Jiazhi Xia, Ying He, Shi-Qing Xin, and Gustavo Patow. 2013. Interactive Applications for Sketch-Based Editable Polycube Map. *© IEEE Transactions on Visualization and Computer Graphics, 2013, vol. 19, núm. 7, p. 1158-1171* (2013).

Michael Garland. 1999. Multiresolution modeling: Survey & future opportunities. *State of the art report* (1999), 111–131.

Bruce Gooch and Amy Gooch. 2001. *Non-photorealistic rendering*. AK Peters/CRC Press.

John C Gower, Garmt B Dijksterhuis, et al. 2004. *Procrustes problems*. Vol. 30. Oxford University Press on Demand.

James Gregson, Alla Sheffer, and Eugene Zhang. 2011. All-hex mesh generation via volumetric polycube deformation. In *Computer graphics forum*, Vol. 30. Wiley Online Library, 1407–1416.

Lei He and Scott Schaefer. 2013. Mesh denoising via L 0 minimization. *ACM Transactions on Graphics (TOG)* 32, 4 (2013), 64.

Ying He, Hongyu Wang, Chi-Wing Fu, and Hong Qin. 2009. A divide-and-conquer approach for automatic polycube map construction. *Computers & Graphics* 33, 3 (2009), 369–380.

Linda Dalrymple Henderson. 1983. *The Fourth Dimension and Non-Euclidean Geometry*. Princeton, Princeton University Press.

Aaron Hertzmann, Carol O'Sullivan, and Ken Perlin. 2009. Realistic human body movement for emotional expressiveness. In *ACM SIGGRAPH 2009 Courses*. ACM, 20.

Hugues Hoppe. 1996. Progressive meshes. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*. ACM, 99–108.

Ruizhen Hu, Wenchao Li, Oliver Van Kaick, Hui Huang, Melinos Averkiou, Daniel Cohen-Or, and Hao Zhang. 2017. Co-locating style-defining elements on 3d shapes. *ACM Transactions on Graphics (TOG)* 36, 3 (2017), 33.

Jin Huang, Tengfei Jiang, Zeyun Shi, Yiying Tong, Hujun Bao, and Mathieu Desbrun. 2014. l1-Based Construction of Polycube Maps from Complex Shapes. *ACM Transactions on Graphics (TOG)* 33, 3 (2014), 25.

Takeo Igarashi, Tomer Moscovich, and John F Hughes. 2005. As-rigid-as-possible shape manipulation. In *ACM transactions on Graphics (TOG)*, Vol. 24. ACM, 1134–1141.

Alec Jacobson, Daniele Panozzo, et al. 2018. libigl: A simple C++ geometry processing library. http://libigl.github.io/libigl/.

Jens Kerber, Art Tevs, Alexander Belyaev, Rhaleb Zayer, and Hans-Peter Seidel. 2009. Feature sensitive bas relief generation. In *2009 IEEE International Conference on Shape Modeling and Applications*. IEEE, 148–154.

Shahar Z Kovalsky, Meirav Galun, and Yaron Lipman. 2016. Accelerated quadratic proxy for geometric optimization. *ACM Transactions on Graphics (TOG)* 35, 4 (2016), 134.

Julian Kratt, Ferdinand Eisenkeil, Sören Pirk, Andrei Sharf, and Oliver Deussen. 2014. Non-realistic 3D Object Stylization. In *Proceedings of the Workshop on Computational Aesthetics (CAe '14)*. ACM, New York, NY, USA, 67–75. https://doi.org/10.1145/2630099.2630102

Jan Eric Kyprianidis, John Collomosse, Tinghuai Wang, and Tobias Isenberg. 2013. State of the "Art": A Taxonomy of Artistic Stylization Techniques for Images and Video. *IEEE transactions on visualization and computer graphics* 19, 5 (2013), 866–885.

Honghua Li, Hao Zhang, Yanzhen Wang, Junjie Cao, Ariel Shamir, and Daniel Cohen-Or. 2013. Curve style analysis in a set of shapes. In *Computer Graphics Forum*, Vol. 32. Wiley Online Library, 77–88.

Isaak Lim, Anne Gehre, and Leif Kobbelt. 2016. Identifying style of 3D shapes using deep metric learning. In *Computer Graphics Forum*, Vol. 35. Wiley Online Library, 207–215.

Juncong Lin, Xiaogang Jin, Zhengwen Fan, and Charlie CL Wang. 2008. Automatic polycube-maps. In *International Conference on Geometric Modeling and Processing*. Springer, 3–16.

Hsueh-Ti Derek Liu, Michael Tao, and Alec Jacobson. 2018. Paparazzi: Surface Editing by way of Multi-View Image Processing. In *SIGGRAPH Asia 2018 Technical Papers*. ACM, 221.

Ligang Liu, Lei Zhang, Yin Xu, Craig Gotsman, and Steven J Gortler. 2008. A local/global approach to mesh parameterization. In *Computer Graphics Forum*, Vol. 27. Wiley Online Library, 1495–1504.

Tiantian Liu, Adam W Bargteil, James F O'Brien, and Ladislav Kavan. 2013. Fast simulation of mass-spring systems. *ACM Transactions on Graphics (TOG)* 32, 6 (2013), 214.

Tianqiang Liu, Aaron Hertzmann, Wilmot Li, and Thomas Funkhouser. 2015. Style compatibility for 3D furniture models. *ACM Transactions on Graphics (TOG)* 34, 4 (2015), 85.

Marco Livesu, Nicholas Vining, Alla Sheffer, James Gregson, and Riccardo Scateni. 2013. PolyCut: monotone graph-cuts for PolyCube base-complex construction. *ACM Transactions on Graphics (TOG)* 32, 6 (2013), 171.

Zhaoliang Lun, Evangelos Kalogerakis, and Alla Sheffer. 2015. Elements of style: learning perceptual shape style similarity. *ACM Transactions on Graphics (TOG)* 34, 4 (2015), 84.

Zhaoliang Lun, Evangelos Kalogerakis, Rui Wang, and Alla Sheffer. 2016. Functionality preserving shape style transfer. *ACM Transactions on Graphics (TOG)* 35, 6 (2016), 209.

Sheng-Jie Luo, Yonghao Yue, Chun-Kai Huang, Yu-Huan Chung, Sei Imai, Tomoyuki Nishita, and Bing-Yu Chen. 2015. Legolization: optimizing LEGO designs. *ACM Transactions on Graphics (TOG)* 34, 6 (2015), 222.

Chongyang Ma, Haibin Huang, Alla Sheffer, Evangelos Kalogerakis, and Rui Wang. 2014. Analogy-driven 3D style transfer. In *Computer Graphics Forum*, Vol. 33. Wiley Online Library, 175–184.

Josiah Manson and Scott Schaefer. 2011. Hierarchical deformation of locally rigid meshes. In *Computer Graphics Forum*, Vol. 30. Wiley Online Library, 2387–2396.

Jacob Mattingley and Stephen Boyd. 2012. CVXGEN: A code generator for embedded convex optimization. *Optimization and Engineering* 13, 1 (2012), 1–27.

Ravish Mehra, Qingnan Zhou, Jeremy Long, Alla Sheffer, Amy Gooch, and Niloy J Mitra. 2009. Abstraction of man-made shapes. In *ACM transactions on graphics (TOG)*, Vol. 28. ACM, 137.

Alessandro Muntoni, Marco Livesu, Riccardo Scateni, Alla Sheffer, and Daniele Panozzo. 2018. Axis-aligned height-field block decomposition of 3d shapes. *ACM Transactions on Graphics (TOG)* 37, 5 (2018), 169.

Yue Peng, Bailin Deng, Juyong Zhang, Fanyu Geng, Wenjie Qin, and Ligang Liu. 2018. Anderson acceleration for geometry optimization and physics simulation. *ACM Transactions on Graphics (TOG)* 37, 4 (2018), 42.

Ulrich Pinkall and Konrad Polthier. 1993. Computing discrete minimal surfaces and their conjugates. *Experimental mathematics* 2, 1 (1993), 15–36.

Michael Rabinovich, Roi Poranne, Daniele Panozzo, and Olga Sorkine-Hornung. 2017. Scalable locally injective mappings. *ACM Transactions on Graphics (TOG)* 36, 2 (2017), 16.

Bernhard Reinert, Tobias Ritschel, and Hans-Peter Seidel. 2012. Homunculus Warping: Conveying importance using self-intersection-free non-homogeneous mesh deformation. In *Computer Graphics Forum*, Vol. 31. Wiley Online Library, 2165–2171.

Christian Schüller, Daniele Panozzo, and Olga Sorkine-Hornung. 2014. Appearance-mimicking surfaces. *ACM Transactions on Graphics (TOG)* 33, 6 (2014), 216.

Liang-Tsen Shen, Sheng-Jie Luo, Chun-Kai Huang, and Bing-Yu Chen. 2012. SD Models: Super-Deformed Character Models. In *Computer Graphics Forum*, Vol. 31. Wiley Online Library, 2067–2075.

Anna Shtengel, Roi Poranne, Olga Sorkine-Hornung, Shahar Z Kovalsky, and Yaron Lipman. 2017. Geometric optimization via composite majorization. *ACM Trans. Graph.* 36, 4 (2017), 38–1.

Wenhao Song, Alexander Belyaev, and Hans-Peter Seidel. 2007. Automatic generation of bas-reliefs from 3d shapes. In *IEEE International Conference on Shape Modeling and Applications 2007 (SMI'07)*. IEEE, 211–214.

Olga Sorkine and Marc Alexa. 2007. As-rigid-as-possible surface modeling. In *Symposium on Geometry processing*, Vol. 4. 109–116.

Oded Stein, Eitan Grinspun, and Keenan Crane. 2018a. Developability of Triangle Meshes. *ACM Trans. Graph.* 37, 4 (2018).

Oded Stein, Eitan Grinspun, Max Wardetzky, and Alec Jacobson. 2018b. Natural Boundary Conditions for Smoothing in Geometry Processing. *ACM Trans. Graph.* 37, 2, Article 23 (May 2018), 13 pages. https://doi.org/10.1145/3186564

Oded Stein, Alec Jacobson, and Eitan Grinspun. 2019. Interactive design of castable shapes using two-piece rigid molds. *Computers & Graphics* (2019).

Marco Tarini, Kai Hormann, Paolo Cignoni, and Claudio Montani. 2004. Polycube-maps. In *ACM transactions on graphics (TOG)*, Vol. 23. ACM, 853–860.

Romain Testuz, Yuliy Schwartzburg, and Mark Pauly. 2013. Automatic Generation of Constructable Brick Sculptures. In *Eurographics 2013 - Short Papers*. The Eurographics Association. https://doi.org/10.2312/conf/EG2013/short/081-084

Christian Theobalt, Christian Roessl, Edilson de Aguiar, and Hans-Peter Seidel. 2007. Animation Collage. In *Eurographics/SIGGRAPH Symposium on Computer Animation*, Dimitris Metaxas and Jovan Popovic (Eds.). The Eurographics Association. https://doi.org/10.2312/SCA/SCA07/271-280

Robert Tibshirani. 1996. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society: Series B (Methodological)* 58, 1 (1996), 267–288.

Andreï Nikolaevitch Tikhonov, AV Goncharsky, VV Stepanov, and Anatoly G Yagola. 2013. *Numerical methods for the solution of ill-posed problems*. Vol. 328. Springer Science & Business Media.

Hongyu Wang, Ying He, Xin Li, Xianfeng Gu, and Hong Qin. 2007. Polycube splines. In *Proceedings of the 2007 ACM symposium on Solid and physical modeling*. ACM, 241–251.

Hongyu Wang, Miao Jin, Ying He, Xianfeng Gu, and Hong Qin. 2008. User-controllable polycube map for manifold spline construction. In *Proceedings of the 2008 ACM symposium on Solid and physical modeling*. ACM, 397–404.

Yu-Shuen Wang, Min-Wen Chao, Chin-Chueng Yi, and Chao-Hung Lin. 2011. Cubist Style Rendering for 3D Polygonal Models. *Journal of Information Science and Engineering* 27, 6 (2011), 1885–1899.

Tim Weyrich, Jia Deng, Connelly Barnes, Szymon Rusinkiewicz, and Adam Finkelstein. 2007. Digital bas-relief from 3D scenes. In *ACM transactions on graphics (TOG)*, Vol. 26. ACM, 32.

Kai Xu, Honghua Li, Hao Zhang, Daniel Cohen-Or, Yueshan Xiong, and Zhi-Quan Cheng. 2010. Style-content separation by anisotropic part scales. In *ACM Transactions on Graphics (TOG)*, Vol. 29. ACM, 184.

Fenggen Yu, Kai Xu, Ali Mahdavi-Amiri, Hao Zhang, et al. 2018. Semi-Supervised Co-Analysis of 3D Shape Styles from Projected Lines. *ACM Transactions on Graphics (TOG)* 37, 2 (2018), 21.

Wuyi Yu, Kang Zhang, Shenghua Wan, and Xin Li. 2014. Optimizing polycube domain construction for hexahedral remeshing. *Computer-Aided Design* 46 (2014), 58–68.

Mehmet Ersin Yumer and Levent Burak Kara. 2012. Co-abstraction of shape collections. *ACM Transactions on Graphics (TOG)* 31, 6 (2012), 166.

Juyong Zhang, Bailin Deng, Hong Yang, Yue Peng, Wenjie Qin, and Ligang Liu. 2018. Static/dynamic filtering for mesh geometry. *IEEE transactions on visualization and computer graphics* (2018).

Hui Zhao, Na Lei, Xuan Li, Peng Zeng, Ke Xu, and Xianfeng Gu. 2018. Robust edge-preserving surface mesh polycube deformation. *Computational Visual Media* 4, 1 (01 Mar 2018), 33–42. https://doi.org/10.1007/s41095-017-0100-x

Qingnan Zhou and Alec Jacobson. 2016. Thingi10K: A Dataset of 10,000 3D-Printing Models. *arXiv preprint arXiv:1605.04797* (2016).

Yufeng Zhu, Robert Bridson, and Danny M Kaufman. 2018. Blended cured quasi-newton for distortion optimization. *ACM Transactions on Graphics (TOG)* 37, 4 (2018), 40.

Denis Zorin. 2006. Modeling with multiresolution subdivision surfaces. In *ACM SIGGRAPH 2006 Courses*. ACM, 30–50.

A POLYHEDRAL GENERALIZATION

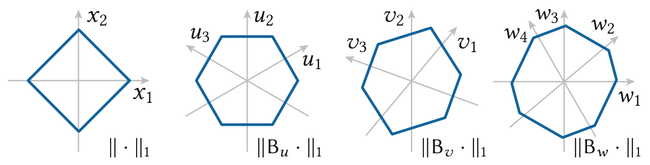

x_2
x_1
$\|\cdot\|_1$
u_3
u_2
u_1
$\|\mathsf{B}_u\cdot\|_1$
v_2
v_3
v_1
$\|\mathsf{B}_v\cdot\|_1$
w_3
w_4
w_2
w_1
$\|\mathsf{B}_w\cdot\|_1$

Fig. 28. By specifying different coordinate transformations B inside the ℓ^1-norm, we can encourage polyhedral style.

Simply applying a coordinate transformation $B : \mathbb{R}^n \to \mathbb{R}^m$ inside the ℓ^1-norm can encourage polyhedral results, instead of cubic results (see Fig. 28). The ℓ^1-norm of a vector is defined as the summation of its magnitudes along each basis vector. Thus applying a coordinate transformation inside the ℓ^1-norm changes its bahavior because the basis vectors are different. Following the notation in Eq. 1, polyhedron energy can be written as

$$\underset{\widetilde{\mathsf{V}},\{\mathsf{R}_i\}}{\text{minimize}} \sum_{i\in V}\sum_{j\in\mathcal{N}(i)} \frac{w_{ij}}{2}\|\mathsf{R}_i\mathsf{D}_{ij} - \widetilde{\mathsf{D}}_{ij}\|_F^2 + \lambda a_i\|\mathsf{B}\mathsf{R}_i\hat{\mathsf{n}}_i\|_1.$$

In our case, B is a m-by-3 coordinate transformation matrix for shapes embedded in $\mathbb{R}^3$. Again by setting $\mathsf{z} = \mathsf{R}_i\hat{\mathsf{n}}_i$ we can reach almost the same optimization procedures, except the Eq. 5 now becomes (we ignore the iteration superscript for clarity)

$$\mathsf{z}^{k+1} \leftarrow \underset{\mathsf{z}}{\arg\min}\ \lambda a_i\|\mathsf{B}\mathsf{z}\|_1 + \frac{\rho}{2}\|\mathsf{R}_i\hat{\mathsf{n}}_i - \mathsf{z} + \mathsf{u}\|_2^2. \tag{10}$$

Similar to common techniques for solving the *Basis Pursuit* problem, we introduce a variable $\mathsf{t} \geq \|\mathsf{B}\mathsf{z}\|_1$ to transform Eq. 10 into a small quadratic program subject to equality constraints

$$\begin{aligned}\underset{\mathsf{z},\mathsf{t}}{\text{minimize}}\ & \begin{bmatrix}\mathsf{z}^\top & \mathsf{t}^\top\end{bmatrix}\begin{bmatrix}\rho/2\cdot\mathsf{I}_3 & 0\\ 0 & 0\end{bmatrix}\begin{bmatrix}\mathsf{z}\\ \mathsf{t}\end{bmatrix}\\ & \quad + \begin{bmatrix}-\rho(\mathsf{R}_i\hat{\mathsf{n}}_i + \mathsf{u})^\top & \lambda a_i \mathsf{1}_m^\top\end{bmatrix}\begin{bmatrix}\mathsf{z}\\ \mathsf{t}\end{bmatrix}\\ \text{subject to}\ & \begin{bmatrix}\mathsf{B} & -\mathsf{I}_m\\ -\mathsf{B} & -\mathsf{I}_m\end{bmatrix}\begin{bmatrix}\mathsf{z}\\ \mathsf{t}\end{bmatrix} \leq 0,\end{aligned}$$

where I_x and $\mathsf{1}_x$ denote the identity matrix with size x and a column vector of 1 with size x respectively. We then solve this efficiently using CVXGEN [Mattingley and Boyd 2012]. Note that the results in Fig. 24 and Fig. 25 use $m = 4$.