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 $\ell^1$-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

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 Serpent 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.
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 \textit{as-rigid-as-possible} (ARAP) energy with an $\ell^1$ 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 [Berkinen et al. 2017; Lun et al. 2016; Ma et al. 2014]. However, these methods are designed for discerning and transferring 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 the collage art [Gal et al. 2007; Theobalt et al. 2007], voxel/lego art [Luo et al. 2015; Testuz et al. 2013], \textit{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; Schuller 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).
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 multi-resolution 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 [Garland 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].

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.

3 METHOD

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

Let V be a |V| × 3 matrix of vertex positions at the rest state and V be a |V| × 3 matrix containing the deformed vertex positions. We denote dij = [vj − vi]T and d̃ij = [vj − vi]T be the edge vectors between vertices i, j at the rest and deformed states respectively.

ACM Trans. Graph., Vol. 38, No. 6, Article 197. Publication date: November 2019.
The energy for our cubic stylization is as follows

\[
\text{minimize}_{\hat{V}, \{R_i\}} \sum_{i \in V} \sum_{j \in N(i)} \frac{w_{ij}}{2} ||R_i d_{ij} - \tilde{d}_{ij}||_2^2 + \lambda a_i ||R_i \hat{n}_i||_1. \tag{1}
\]

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 \(N(i)\) denotes the "spokes and rims" edges of the \(i\)th 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 \(\lambda\) to exhibit the similar cubeness across different mesh resolutions. Intuitively, minimizing the \(\ell^1\)-norm of the rotated normal encourages \(R_i \hat{n}_i\) to align with one of coordinate axes because \(\ell^1\)-norm encourages sparsity. Combining the two, the optimal rotation \(\{R_i^*\}\) 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 \(V\), 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 \(\ell^1\) term.

### 3.1 Local-Step

Our local step for each vertex \(i\) can be written as

\[
R_i^* = \arg \min_{R_i \in SO(3)} \frac{1}{2} ||R_i D_i - \tilde{D}_i||_2^2 + \lambda a_i ||R_i \hat{n}_i||_1, \tag{2}
\]

where \(W_i\) is a \(|N(i)| \times |N(i)|\) diagonal matrix of cotangent weights, \(D_i\) and \(\tilde{D}_i\) are \(3 \times |N(i)|\) matrices of rim/spoke edge vectors at the rest and deformed states respectively. We denote \(||X||_2^2 = \text{Tr}(X^T X)\) for notational convenience. By setting \(z = R_i \hat{n}_i\), we can rewrite Eq. 2 as

\[
\text{minimize}_{z, R_i \in SO(3)} \frac{1}{2} ||R_i D_i - \tilde{D}_i||_2^2 + \lambda a_i ||z||_1 \tag{3}
\]

subject to \(z = R_i \hat{n}_i = 0\).

Fig. 9. One can control the cubeness by changing the \(\lambda\) parameter in Eq. 1.

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]:

\[
R_i^{k+1} \leftarrow \arg \min_{R_i \in SO(3)} \frac{1}{2} ||R_i D_i - \tilde{D}_i||_2^2 + \frac{\rho}{2} ||R_i \hat{n}_i - z^k + u^k||_2^2 \tag{4}
\]

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

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

\[
\rho^{k+1} \leftarrow \text{update}\left(\rho^k\right) \tag{7}
\]

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]

\[
R_i^{k+1} \leftarrow \arg \max_{R_i \in SO(3)} \text{Tr}(R_i M_i) \tag{8}
\]

where \(M_i = [D_i \ \hat{n}_i] [W_i \ \rho^k] \left(\tilde{D}_i^T \ (z^k - u^k)\right)\).

One can derive the optimal \(R_i\) from the singular value decomposition of \(M_i = U_i \Sigma_i V_i^T\):

\[
R_i^{k+1} \leftarrow V_i U_i^T, \tag{8}
\]

up to changing the sign of the column of \(U_i\) so that \(\det(R_i) > 0\).
We update the penalty with zeros, and set the initial $u$, and different cubenesses in Fig. 9 (right). In Fig. 11, we also turn meshes with boundaries (red) into the cubic style. 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 implies 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:

$$z^{k+1} \leftarrow S_{\lambda \rho / \lambda_0} (Rz^{k+1} + u^k)$$

We update the penalty $\rho$ (Eq. 7) according to Sec. 3.4.1 in [Boyd et al. 2011]. Then $z$, $u$, and $\rho$ are reused in consecutive iterations. Note that for extremely large $\lambda$ one may need to increase the initial value of $\epsilon^{\text{abs}}$ accordingly in order to avoid local minima. We stop the optimization when the relative displacement, the infinity norm of relative per vertex displacements, is lower than $3 \times 10^{-3}$ (see Fig. 12 for the convergence plots). More elaborate stopping criteria, such as the method of [Zhu et al. 2018], could also be used.

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 the mesh to lead to different results. Fig. 13. The global orientation of the shape influences the $\ell^1$ term in Eq. 1. Applying different rotations to the mesh lead to different results.

Algorithm 1: Cube Stylization ($\lambda$)

**Input**: A triangle mesh $V, F$

**Output**: Deformed vertex positions $\tilde{V}$

1. $\tilde{V} \leftarrow V$
2. while not converge do
3. \quad $R \leftarrow \text{local-step}(V, \tilde{V}, \lambda)$
4. \quad $\tilde{V} \leftarrow \text{global-step}(R)$

Algorithm 2: Fast Cube Stylization ($\lambda, m$)

**Input**: A triangle mesh $V, F$

**Output**: Deformed vertex positions $\tilde{V}$

// pre-processing
1. $m \leftarrow$ target number of faces
2. $V_c, F_c \leftarrow \text{edge collapses}(V, F, m)$

// cubic stylization
3. $\tilde{V}_c \leftarrow V_c$
4. while not converge do
5. \quad $R \leftarrow \text{local-step}(V_c, \tilde{V}_c, \lambda)$
6. \quad $\tilde{V}_c \leftarrow \text{global-step}(R)$
7. $\tilde{V}, F \leftarrow \text{affine vertex splits} (\tilde{V}_c, F_c)$

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 $\lambda$ to be a design parameter controlling the cubeness of a shape (Fig. 9).

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 $\lambda$.

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 $A$:

$$A = (Q_i Q_i^T)^{-1} Q_i,$$

$Q_i$ is a $3 \times |N(i)|$ matrix where each column is the vector from $p_i$ to one of its one-rings neighbors $N(i)$. If $(Q_i Q_i^T)$ 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

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

where $\bar{p}_i$ denotes the position of vertex i in the cubified coarsened shape, and $Q_i$ is a $3 \times |N(i)|$ matrix containing vectors from $\bar{p}_i$ 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 $\lambda$ 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 $\lambda$, 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 $m$.

### 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 $\lambda$ in Fig. 9 and 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 $\lambda$. 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 ($|F|$), $l_1$ weight ($\lambda$), number of faces of the coarsest mesh ($m$), number of iterations (Iters), pre-processing time (Pre.), and runtime at the online stage (Runtime).

| Model         | $|F|$ | $\lambda$ | $m$ | 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  | 0.40      | n/a | 86    | n/a  | 2.26s   |
| Fig. 16, left | 2018K| 0.20      | 20K | 83    | 64.19s| 3.93s   |
| Fig. 16, mid. | 346K | 0.40      | 20K | 222   | 10.69s| 4.59s   |
| Fig. 16, right| 811K | 0.30      | 40K | 173   | 30.44s| 8.38s   |

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 $\lambda, m$, 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 $\lambda$. Although Eq. 1 only has a single $\lambda$ for an entire shape, we can actually specify different $\lambda_i$ for each vertex independently to have non-uniform cubeness, which leads to the expression $\lambda_i a_i ||R_i \hat{n}_i||_1$. 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 $\lambda_i$ 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 $a_i (\lambda_x ||R_i \hat{n}_i||_x + \lambda_y ||R_i \hat{n}_i||_y + \lambda_z ||R_i \hat{n}_i||_z)$ and specify different values for each $\lambda_x, \lambda_y, \lambda_z$ to have the style of a rectangular prism.

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 $V_i$ at user-desired positions. We can also use the same methodology to
Constrain some parts of the geometry lying on certain planes. For instance, setting \((V_i)_x = 0\) 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 \textit{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 \(\lambda a_i \parallel R_i \hat{n}_{\text{local}} \parallel_1\), where we use \(\hat{n}_{\text{local}}\) 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 \(\lambda a_i \parallel BR_i \hat{n}_{\text{local}} \parallel_1\) 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).

5 LIMITATIONS & FUTURE WORK

Accelerating the stylization to real-time would enable faster iterations between designs. Developing a more robust stylization tool 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, \textit{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, Youssef Alami Mejjati, and Zhicong Lu for participating in the user study.

REFERENCES

Sema Berkitten, 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 Schwartzberg, 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. Convex optimization. © 2011, The MathWorks Inc. http://www.mathworks.com/products/optimization/.

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.

Jianyu Li, 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.

Jie Liu, Xinghao Jin, Zhengwen Fan, and Charlie CL Wang. 2008. Automatic polycube-maps. In International Conference on Geometric Modeling and Processing. Springer, 3–16.

Ligang Liu, Lei Zhang, Yin Xu, Craig Gotsman, and Steven J Gortler. 2007. A local/global approach to mesh parameterization. In Computer Graphics Forum. Vol. 26. ACM, 1134–1141.

Zhimao Lu for participating in the user study.

Lei He and Scott Schaefer. 2013. Mesh denoising via L0 minimization. ACM Transactions on Graphics (TOG) 32, 4 (2013), 64.

Ying He, Hongyu Wu, 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.

Jan Hwang, Tengfei Jiang, Zeyun Shi, Tiying Tong, Hujun Bao, and Mathieu Desbrun. 2014. 1¹-Based Construction of Polycube Maps from Complex Shapes. ACM Transactions on Graphics (TOG) 33, 4 (2014), 25.

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, Hua Huang, Melamos 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.

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.

ACM Trans. Graph., Vol. 38, No. 6, Article 197. Publication date: November 2019.
A POLYHEDRAL GENERALIZATION

Fig. 28. By specifying different coordinate transformations B inside the ℓ1-norm, we encourage polyhedral style.

Simply applying a coordinate transformation \( B : \mathbb{R}^n \rightarrow \mathbb{R}^m \) inside the ℓ1-norm can encourage polyhedral results, instead of cubic results (see Fig. 28). The ℓ1-norm of a vector is defined as the summation of its magnitudes along each basis vector. Thus applying a coordinate transformation inside the ℓ1-norm changes its behavior because the basis vectors are different. Following the notation in Eq. 1, polyhedron energy can be written as

\[
\min_{V, \{R_i\}} \sum_{i \in V} \sum_{j \in N(i)} \frac{wij}{2} \| R_i D_{ij} - \hat{D}_{ij} \|_2^2 + \lambda a_i \| B R_i \hat{n}_i \|_1.
\]

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

\[
z^{k+1} \leftarrow \arg \min \lambda a_i \| B z_i \|_1 + \frac{p}{2} \| R_i \hat{n}_i - z + u \|_2^2.
\]  

(10)

Similar to common techniques for solving the Basis Pursuit problem, we introduce a variable \( t \geq \| B z_i \|_1 \) to transform Eq. 10 into a small quadratic program subject to equality constraints

\[
\min_{z, t} \begin{bmatrix} z^T & t^T \end{bmatrix} \begin{bmatrix} p/2 & I_3 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} z \\ t \end{bmatrix} + \left[ -\rho (R_i \hat{n}_i + u)^T \lambda a_i T_m \right] \begin{bmatrix} z \\ t \end{bmatrix}
\]

subject to

\[
\begin{bmatrix} t \\ -B \end{bmatrix} \begin{bmatrix} I_m \\ -I_m \end{bmatrix} \begin{bmatrix} z \\ t \end{bmatrix} \leq 0,
\]

where \( I_m \) and \( I_3 \) 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 \( m = 4 \).