CubicCraft: A Mesh Stylization Tool

Haoda Li
haoda_li@berkeley.edu
University of California Berkeley
Berkeley, California, USA

Puyuan Yi
yipuyuan@berkeley.edu
University of California Berkeley
Berkeley, California, USA

Weiji Li
weiji@berkeley.edu
University of California Berkeley
Berkeley, California, USA

Zhen Jiang
zhen_jiang16@berkeley.edu
University of California Berkeley
Berkeley, California, USA

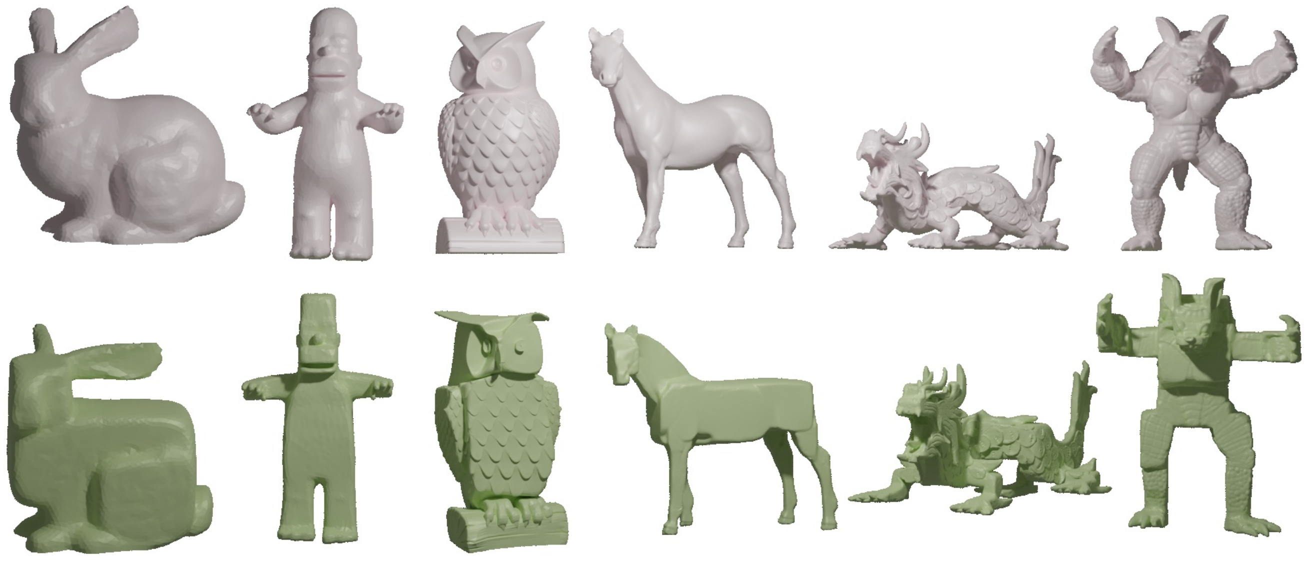


Figure 1: Cubic Craft turns triangle meshes (grey) into cubic-styled meshes (green)

ABSTRACT

We present a stylization tool to automatically manipulate triangle meshes into a cubic style. Our tool uses a cubic stylization algorithm[Liu and Jacobson 2019] to cubify the user's provided meshes. The algorithm extends the as-rigid-as-possible energy [Sorkine and Alexa 2007] with an additional L1 regularization, hence can work seamlessly with ADMM optimization. Cubic stylization works only on the vertex positions, hence preserving the geometrical details and topology. In addition, we implemented the algorithm with GPU acceleration and achieves real-time interactive editing. We also created a user-friendly interaction surface to let users easily change the algorithm's hyperparameter and cubify their own mesh. With our tool, 3D artists can create Minecraft-styled objects with ease. Our code is available at https://github.com/haoda-li/CS284A-cubic-craft.

CCS CONCEPTS

• **Computing methodologies → Mesh geometry models**; Parallel algorithms; • **Human-centered computing** → *Visualization toolkits.*

KEYWORDS

mesh stylization, mesh deformation, geometry processing, visualization tools

ACM Reference Format:
Haoda Li, Puyuan Yi, Weiji Li, and Zhen Jiang. 2023. CubicCraft: A Mesh Stylization Tool. In *Proceedings of Make sure to enter the correct conference title from your rights confirmation emai (Conference acronym 'XX).* ACM, New York, NY, USA, 5 pages. https://doi.org/XXXXXXX.XXXXXXX

1 INTRODUCTION

With the increasing availability of image stylization filters and non-photorealistic rendering techniques, creating artistic images has become much more accessible to non-professional users. However, the direct stylization of 3D shapes and non-realistic modeling has not yet been given as much attention. Despite the advancements in technology, professional industries like visual effects and video games still rely on trained modelers to meticulously create non-realistic geometric assets. This is because exploring geometric styles presents a greater challenge, as it involves dealing with arbitrary topologies, curved metrics, and non-uniform discretization. While image stylization tools have made it easier to generate artistic imagery, there is still a lack of effective tools for generating artistic geometry, which remains a major obstacle to the development of geometric stylization.

The focus of this paper is on a specific style of sculpture, namely the cubic style. This style has been prevalent throughout art history (ancient sculptures) and modern game history (Minecraft). In this work, we have developed a stylization tool *cubic stylization* based on GPU that takes a 3D shape as input and outputs a deformed shape that has the same style as cubic sculptures. This tool is aimed at helping artists and designers achieve the cubic style more easily and efficiently, while also providing a new way to explore and experiment with this timeless artistic tradition.

Our implemented method *cubic stylization* formulates the task as an energy optimization problem, which preserves the geometric details of a shape while transforming it into a cubic form. Specifically, this energy function combines an as-rigid-as-possible (ARAP) energy with a special L1 regularization. This energy can be minimized efficiently using the local-global approach with the Alternating Direction Method of Multipliers (ADMM). This method has strong flexibility that allowing artists and designers to achieve a wide range of stylistic variations within the cubic style, providing them with greater creative freedom and expressive potential.

Our main contributions are summarized as follows:

- We implemented GPU-accelerated *cubic stylization* algorithm, which allows real-time cubic style object generation and rendering.
- We created a user-friendly GUI to interact with our algorithm. Users can easily change the hyperparameter of the algorithm and observe different results.

2 RELATED WORKS

In this section, our primary focus is on exploring methods for processing geometry. Specifically, we will be discussing various techniques for studying geometric styles and deformation methods that share common technical similarities. Our aim is to provide a comprehensive overview of these methods and their applications and to highlight their significance in the field of geometry processing.

2.1 Shape Deformation

The subfield of shape deformation has been the subject of considerable research in computer graphics and related fields. Several notable works have contributed to this area, including the as-rigid-as-possible (ARAP) energy model [Sorkine and Alexa 2007], which has become a popular method for shape deformation due to its ability to preserve the rigid structure of the object being deformed. Another important contribution is the Laplacian-based deformation technique [Sorkine et al. 2004], which uses the Laplacian operator to deform a shape while preserving its surface details. In addition, several works have explored the use of physically-based deformation models, such as the finite-element method and the mass-spring system [Nealen et al. 2006].

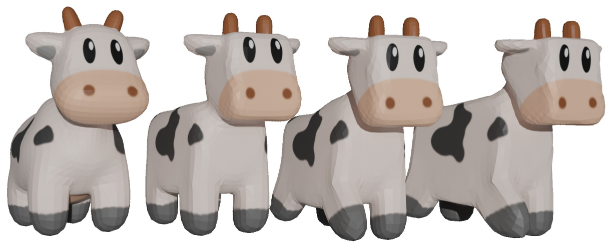


Figure 2: Shape deformation preserves vertex attributes

More recent work in shape deformation has focused on extending these techniques to handle more complex shapes and deformation scenarios. For example, the Cage-based deformation method [Le and Deng 2017] uses a cage mesh to define the deformation space, allowing for more intuitive and flexible deformation. Other works have explored the use of machine learning techniques, such as neural networks, for shape deformation [Li et al. 2022].

The subfield of shape deformation has made significant contributions to the field of computer graphics and related disciplines and continues to be an active area of research with numerous exciting directions for future exploration. Our cubic stylization is also a specific kind of shape deformation that combines ARAP energy term and a L1 regularization term.

2.2 Different Geometric Style

Research on different geometric styles has been a topic of interest in computer graphics and related fields. Two main approaches have been taken to explore this area: discriminative geometric styles and generative geometric styles.

Discriminative geometric styles focus on identifying and analyzing different styles in existing geometry. For example, the work by Kim et al. [Kim et al. 2013] proposes a method for identifying and characterizing different geometric styles in furniture design, while the work by Zhou et al. [Zhou et al. 2016] focuses on identifying different styles in fashion design.

On the other hand, generative geometric styles aim to create new geometry in a particular style. One notable example is the work by Huang et al. [Huang et al. 2018], which proposes a method for generating 3D models in a particular style using a generative adversarial network (GAN). Another example is the work by Kalogerakis et al. [Kalogerakis et al. 2012], which uses a probabilistic model to generate 3D shapes with a particular style.

Overall, research on different geometric styles has led to a deeper understanding of the principles and characteristics of different styles in various domains, as well as the development of methods for creating new geometry in a particular style. These techniques

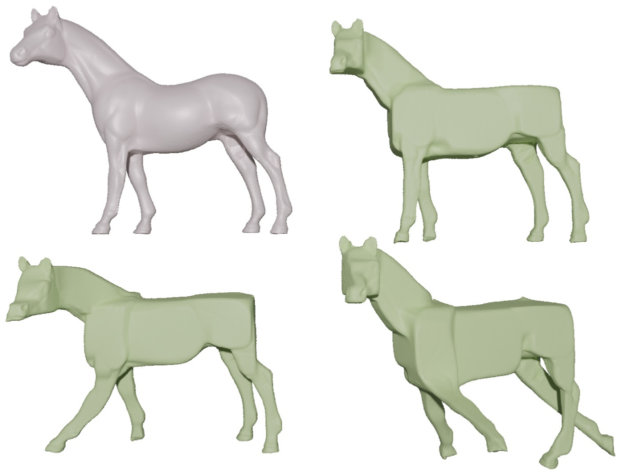
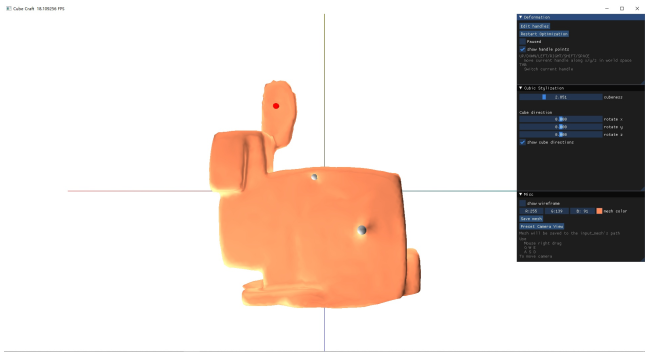


Figure 3: Cubified Mesh with as-rigid-as-possible deformation



```
[ ] Paused
[✓] show handle points
UP/DOWN/LEFT/RIGHT/SHIFT/SPACE
  move current handle along x/y/z in world space
TAB
  Switch current handle
2.851 cubeness
Cube direction
0.000 rotate x
0.000 rotate y
0.000 rotate z
[✓] show cube directions
[ ] show wireframe
R:255  G:139  B: 91  mesh color
Mesh will be saved to the input_mesh's path
Use
  Mouse right drag
  Q W E
  A S D
To move camera
```

Figure 4: Our GUI allows users to tune parameters and perform deformation

have potential applications in a variety of fields, such as architecture, product design, and entertainment.

3 METHOD

Our method is based on cubic stylization [Liu and Jacobson 2019], a method to deform the input mesh into a cubic stylized mesh. Generally, the method adds a new L1 regularization on the deformation with As-rigid-as-possible (ARAP) [Sorkine and Alexa 2007]energy optimization. By regularizing each vertex's normals to align with the axis, the mesh can have a cubic style, while maintaining the local geometric details.

The problem description is illustrated as follows: given a triangle mesh $S = (V, F)$ and a set of constraints on vertex positions. We want to output a deformed shape $\tilde{V}$. The output shape will have each sub-component in the style of axis-aligned cubes and will retain the geometric details of the original mesh.

We will describe our method and implementation in the following sections: In section 3.1, we will talk about the As-rigid-as-possible (ARAP) Deformation [Sorkine and Alexa 2007], and elaborate on its energy functions. Then we will talk about cubic Stylization [Liu and Jacobson 2019] in section 3.2. The implementation part is in section 3.3.

3.1 As-rigid-as-possible Deformation

The thought of ARAP energy is very intuitive: given the cell C_i corresponding to vertex i, and its deformed version $\tilde{C}_i$, ARAP defines the approximate rigid transformation between the two cells by observing the edges emanating from the vertex i in S and $\tilde{S}$, where S and $\tilde{S}$ denote the original triangle mesh and the deformed triangle mesh. Note that $\tilde{S}$ should have the same connectivity as S. If the deformation $C_i \longrightarrow \tilde{C}_i$ is rigid, there must exists a rotation matrix R_i such that:

$$\tilde{V}_i - \tilde{V}_j = R_i(V_i - V_j), \forall j \in \mathcal{N}(i) \tag{1}$$

$\mathcal{N}(i)$ denotes the set of vertices connected to vertex i, also called the one-ring neighbors.

When the deformation is not rigid, we can still find the best approximating rotation matrix R_i that fits the above equations in a weighted least squares sense, i.e., minimizes

$$E(C_i, \tilde{C}_i) = \sum_{i \in V} \sum_{j \in \mathcal{N}(i)} w_{ij} \| R_i d_{ij} - \tilde{d_{ij}} \|_F^2 \tag{2}$$

where w_{ij} is the cotangent weight [Pinkall and Polthier 1993] between vertex i and vertex j, and $d_{ij} = V_i - V_j$ and $\tilde{d_{ij}} = \tilde{V}_i - \tilde{V}_j$. What we need is to solve for vertex position $\tilde{V}_i$ and per-vertex rotations R_i that minimizes the energy function above.

For deformation, we are given user-defined constraints on some vertex positions and we need to update all other vertices to minimize the energy. Sorkine and Alexa uses alternating minimization strategy. For each iteration, we first fix vertex positions to find the optimal rotations for each vertex, and then fix the rotations to update vertex positions. The rotation updates only depends on the one-ring neighbor for each vertex, hence we call it a local step. We will talk more details about local steps in the next section. The vertices update, or the global step, can be directly derived by setting the partial derivative w.r.t. each vertex position to 0. Eventually, we need to solve a system of $3N$ equations of $3N$ unknowns, where each vertex corresponds to the equation

$$\sum_{j \in \mathcal{N}(i)} w_{ij} \tilde{d_{ij}} = \sum_{j \in \mathcal{N}(i)} \frac{w_{ij}}{2} (R_i + R_j) d_{ij} \tag{3}$$

3.2 Cubic Stylization

In this section, we will illustrate the cubic stylization algorithm. Intuitively, an object is cubic style if its normals are aligned with the three dominant directions. Therefore, Liu and Jacobson proposed an additional L1 regularization term on the rotated normal. Combining with the ARAP energy, the full energy term is listed as follows:

$$E(C_i, C_i') = \sum_{i \in V} \sum_{j \in \mathcal{N}(i)} \frac{w_{ij}}{2} \| R_i d_{ij} - \tilde{d}_{ij} \|_F^2 + \lambda a_i \| R_i \tilde{n}_i \|_1 \tag{4}$$

In the L1 regularization term, $\hat{n}_i$ denotes the area-weighted unit normal vector of v_i and a_i is the barycentric area of v_i. and λ is the "cubeness" parameter.

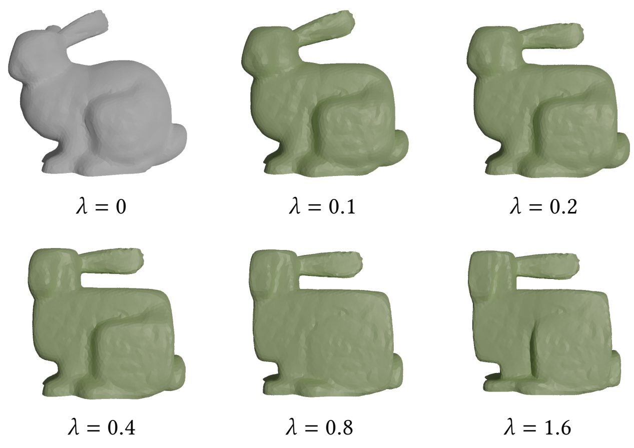


Figure 5: Meshes with different cubeness

The local step involves finding the rotation matrix $R_1, \cdots, R_n$, for each vertex i, we are to optimize:

$$R_i^* = \arg \min_{R_i \in SO(3)} \sum_{j \in \mathcal{N}(i)} \frac{w_{ij}}{2} \|R_i d_{ij} - \tilde{d}_{ij}\|_F^2 + \lambda a_i \|R_i \tilde{n}_i\|_1 \quad (5)$$

note that the ARAP energy can be expressed in matrix formations

$$\frac{1}{2}(R_i D_i - \tilde{D}_i)^T W_i (R_i D_i - \tilde{D}_i) = \frac{1}{2}\|R_i D_i - \tilde{D}_i\|_{W_i}^2 \quad (6)$$

where $D_i, \tilde{D}_i \in \mathbb{R}^{3 \times |\mathcal{N}(i)|}$ are stacked rim/spoke edge vectors and W_i is the diagonal matrix of $w_1, ..., w_n$. Then, write $z = R_i \hat{n}_i$, we can turn the formation into

$$\text{minimize}_{z, R_i} \quad \frac{1}{2}\|R_i D_i - \tilde{D}_i\|_{W_i}^2 + \lambda a_i \|z\|_1 \quad (7)$$

$$\text{subject to} \quad z - R_i \hat{n}_i = 0 \quad (8)$$

Now We can solve the local step using the alternating direction method of multipliers (ADMM) updates [Boyd et al. 2011]. Applying ADMM, the update steps are

$$R_i^{k+1} = \arg\min \frac{1}{2}\|R_i D_i - \tilde{D}_i\|_{W_i}^2 + \frac{\rho^k}{2}\|R_i \hat{n}_i - z^k + u^k\|_2^2 \quad (9)$$

$$z^{k+1} = \arg\min \lambda a_i \|z\|_1 + \frac{\rho^k}{2}\|R_i^{k+1}\hat{n}_i - z + u^k\|_2^2 \quad (10)$$

$$\tilde{u}^{k+1} = u^k + R_i^{k+1}\hat{n}_i - z^{k+1} \quad (11)$$

$$\rho^{k+1}, u^{k+1} = \text{update}(\rho^k) \quad (12)$$

Then, consider each update, The rotation update can be viewed as

$$R_i^{k+1} = \arg\max tr(R_i M_i) \quad (13)$$

$$M_i = \begin{bmatrix} [D_i] & [\hat{n}_i] \end{bmatrix} \begin{bmatrix} [W_i] & 0 \\ 0 & \rho^k \end{bmatrix} \begin{bmatrix} [\tilde{D}_i] \\ [(z^k - u^k)^T] \end{bmatrix} \quad (14)$$

This becomes an Orthogonal Procrustes problem, and the solution is given through single value decomposition

$$M = U\Sigma V^T, R = UV^T \quad (15)$$

Mesh name	|V|	CPU time (s)	GPU time (s)
homer	6002	10.03	3.39
bunny	6172	27.56	3.55
owl	39416	160.52	6.69
horse	48485	211.62	8.25
armadillo	49990	217.96	7.80
dragon	62472	335.71	9.16

Table 1: Running time for Cubic Stylization

up to $\det(R) > 0$ by alternating the sign of U's column. The z update is an instance of lasso problem, which can be solved with a shrinkage step

$$z^{k+1} = S_{\lambda a_i/\rho^k}(R_i^{k+1}\hat{n}_i + u^k) \quad (16)$$

where the shrinkage is defined as

$$S_\chi(x_j) = (1 - \frac{\chi}{|x_j|}) + x_j \quad (17)$$

Hence we solve the local step. Then, we notice that L1 term $\lambda a_i \|R_i \tilde{n}_i\|_1$ is independent of the vertex positions V. Therefore, the global step is exactly the same as ARAP energy optimization.

3.3 Implementation

We implement the cubic stylization [Liu and Jacobson 2019] algorithm using Python and LIBIGL [Jacobson et al. 2018]. We follow Liu and Jacobson's implementation and set the initial $\rho = 10^{-4}, \mu = 50, \tau = 2$. In addition, we observe that the local step updates each vertex independently, providing opportunities for parallelization. We use TAICHI [Hu et al. 2019] to implement a GPU-accelerated version. To maximize parallelism, for each local step, we run a fixed number of ADMM iterations instead of using the stopping criteria. We set the initial ADMM iterations to 50 and reduce it to 5 through the steps. From experiments, we found that this strategy is adequate for convergence.

Compared to the CPU implementation [Liu and Jacobson 2019], our implementation gradually accelerated the local step computation. We tested our implementation on an AMD R9 5900HS CPU with a NVIDIA 3050ti GPU and listed the performance in Table 1.

4 USER INTERFACE

We provide a graphical interface for the users to visualize and easily edit the meshes. The graphical interface is based on the GUI system provided by TAICHI. Given a triangle mesh, our graphical interface allows the user to change the parameters in the algorithm, visualize the deformations, and save the resulting mesh.

In our GUI, the user can directly change the 'cubeness' parameter and observe different results in real-time. The example of different cubic stylization parameters that work differently on bunny.obj is shown in Figure 5. Note that our algorithm only changes the vertex position, hence other local geometric information, such as texture coordinates, is preserved. As shown in Figure 2, the UV texture is preserved through deformation. In addition to the cubeness parameter, we notice that cube stylization is orientation dependent. The cubeness is achieved by forcing all vertex normals to align with

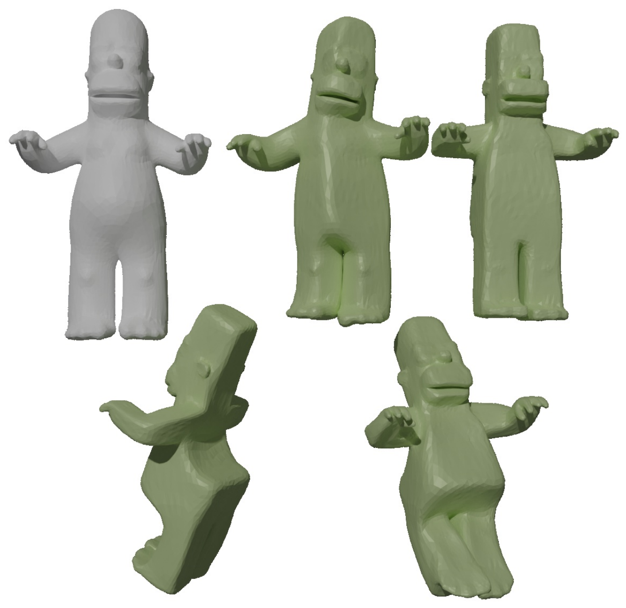


Figure 6: Meshes with different cube orientation

the three standard axes. If we rotate the input mesh, the output shape will be different. Note that the same effect can be achieved by applying a coordinate transformation on all vertex normals. Therefore, we add the coordinate rotation parameters so that users can have different cube orientations. The experiments of different orientation on cubic stylization are presented in Figure 6.

Similar to Sorkine and Alexa's approach, we can put constraints on vertex positions. Users can utilize our GUI to add handle points and move the handle points to perform as-rigid-as-possible deformation. We present some typical deformation results obtained with this technique in Figure 3. Note that natural deformations are obtained because the optimization automatically produces the correct local rotations for each vertex. Our interface example is visualized in Figure 4. The example mesh is Stanford Bunny and the red dots are the user's added constrained points. Our full demo video is available at https://github.com/haoda-li/CS284A-cubic-craft.

5 CONCLUSION

In conclusion, our work presents a powerful tool for cubic stylization that enables 3D artists to create Minecraft-styled objects with ease. Our algorithm, which extends the as-rigid-as-possible energy with an L1 regularization, works seamlessly with ADMM optimization and preserves the underlying geometrical details and topology of the mesh. Furthermore, our implementation with GPU acceleration allows for real-time interactive editing, making the tool both efficient and intuitive to use.

Overall, our work contributes to the growing field of geometry processing by presenting a novel approach to stylization. The ability to manipulate and transform meshes in a cubic style has significant potential for a range of applications, including architectural design, game development, and animation. We believe that our tool will be particularly valuable to 3D artists who wish to create unique and visually striking objects quickly and efficiently.

REFERENCES

Stephen Boyd, Neal Parikh, Eric Chu, Borja Peleato, and Jonathan Eckstein. 2011. Distributed Optimization and Statistical Learning via the Alternating Direction Method of Multipliers. *Found. Trends Mach. Learn.* 3, 1 (jan 2011), 1–122. https://doi.org/10.1561/2200000016

Yuanming Hu, Tzu-Mao Li, Luke Anderson, Jonathan Ragan-Kelley, and Frédo Durand. 2019. Taichi: a language for high-performance computation on spatially sparse data structures. *ACM Transactions on Graphics (TOG)* 38, 6 (2019), 201.

Yifei Huang, Hui Li, Xiaoshuai Sun, and Yongjian Wu. 2018. 3D model generation in a specific style using GAN. In *Proceedings of the 2018 ACM Multimedia Conference.* Association for Computing Machinery (ACM), 1648–1656. https://doi.org/10.1145/3240508.3240664

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

Evangelos Kalogerakis, Holger Winnemoeller, and Jitendra Malik. 2012. Probabilistic reasoning for assembly-based 3D modeling. *ACM Transactions on Graphics (TOG)* 31, 4 (2012), 51. https://doi.org/10.1145/2185520.2185578

HJ Kim, SH Jung, S Lee, and K Lee. 2013. Identification and characterization of different geometric styles in furniture design. *International Journal of Advanced Manufacturing Technology* 69, 5-8 (2013), 1079–1093.

Binh Huy Le and Zhigang Deng. 2017. Interactive cage generation for mesh deformation. In *Proceedings of the 21st ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games* (San Francisco, USA) *(I3D '17)*. Association for Computing Machinery, New York, NY, USA, 1–9.

Jichao Li, Xiaosong Du, and Joaquim R.R.A. Martins. 2022. Machine learning in aerodynamic shape optimization. *Progress in Aerospace Sciences* 134 (2022), 100849. https://doi.org/10.1016/j.paerosci.2022.100849

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

Andrew Nealen, Matthias Müller, Richard Keiser, Eddy Boxerman, and Mark Carlson. 2006. Physically Based Deformable Models in Computer Graphics. *Comput. Graph. Forum* 25 (12 2006), 809–836. https://doi.org/10.1111/j.1467-8659.2006.01000.x

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

Olga Sorkine and Marc Alexa. 2007. As-Rigid-as-Possible Surface Modeling. In *Proceedings of the Fifth Eurographics Symposium on Geometry Processing* (Barcelona, Spain) *(SGP '07)*. Eurographics Association, Goslar, DEU, 109–116.

Olga Sorkine, Daniel Cohen-Or, Yaron Lipman, Marc Alexa, Christian Rössl, and Hans-Peter Seidel. 2004. Laplacian Surface Editing. In *Proceedings of the 2004 Eurographics/ACM SIGGRAPH symposium on Geometry processing* (Nice, France) *(SGP '04)*. Association for Computing Machinery, New York, NY, USA, 175–184.

Ying Zhou, Bin Xu, Yu Guo, and Yong Liu. 2016. Fashion style identification based on convolutional neural networks. In *Proceedings of the 2016 ACM on Multimedia Conference.* Association for Computing Machinery (ACM), 682–686. https://doi.org/10.1145/2964284.2967255