Seminar Report on Cubic Stylization

Meng Liu

Robotics, Cognition, Intelligence Technische Universität München

Abstract

In this report, a paper Cubic Stylization [4] from Hsueh-Ti Derek Liu and Alec Jacobson will be discussed in detail. Published in year 2019, this work proposes a novel algorithm to turn a 3D shape into a style of a cube and preserve the detail of the original shape at the same time. To achieve the goal, as-rigid-as-possible and $\ell 1$-norm are introduced to build up the objective function; ADMM (alternating direction method of multipliers) and affine progressive mesh are adapted to solve the optimization problem efficiently. Some sophisticated artistic controls are provided and discussed at the end of this report.

1 Introduction

In computer graphics and interactive design, cubic stylization means convert a 3D object into a shape of a cube. However, this modeling has a quite long history in the art domain. From the exhibition of Fondation Giacometti in Paris to private collection of Tai-chi series statue; from crystal sculpture to wood carving, shown in figure 1, this geometric modeling is quite widespread across the world and time.

Figure 1: Alberto Giacometti, Personnage accroupi, Fondation Giacometti, Paris (left). Taichi Series: Thrust, Zhu Ming, private collection (upper middle). Crystal cubic sculpture from an online store Sweden (lower middle). Azande Wood Post, DRC, African Wood Carvings (right).

Not only art creation, the need of cubic stylization also come from industrial, e.g. film visual effects. That's why it is also naturally to think about simulating and speeding up the stylization process with a computer when a non-cubified object at hand. In this report, the core idea and the implementation of the paper [4] will be presented in the following steps:

- two building blocks of the objective function will be introduced in section 2.1.1 and 2.1.2 respectively,
- the update rules of ADMM for solving the optimization problem is derived from scratch in section 2.2.1,
- the adaptation of Affine Progressive Method to shorten the computing time is introduced in section 2.2.2,
- and some artistic controls which provided by the model to make the stylization more flexible and meet the need of modelers are presented in section 3.

2 Method description

Comparing to previous works, which output a 2D image, this paper focuses on a more general setting. Given a 3D object in the form of triangular mesh, after processing, it should provide a similar 3D object but in cubic shape. Intuitively, we should let as many faces as possible of a triangular mesh orthogonal to one of the axes of the global coordinate. We can use the normals of a triangular mesh to simplify this, which means the normals should be parallel to one of global coordinate axes. These kinds of vector share one of the forms, $(x, 0, 0)$, $(0, y, 0)$ or $(0, 0, z)$, in which most of coordinates are zeros expect one. These sparse vectors lead us to the first part of the objective.

2.1 Objective Function

2.1.1 Sparsity effect of the ℓ1-norm

Sparsity effect of ℓ1-norm has been explored extensively from machine learning, control theory or any other domains which uses optimization. Even though this property has been of benefit to wild range of sciences without doubt, it is also beneficial to ask why ℓ1-norm leads to sparsity. As we can see from the figure 2 here. Suppose the red line is an object function in 2D plane,

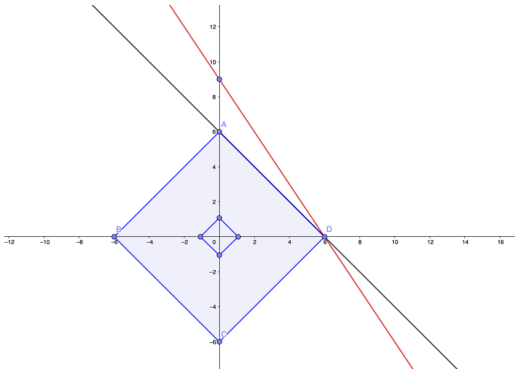

Figure 2: A set circles of ℓ1-norm

Figure 3: Unit spheres of different norms[2].

we need to find the optimal solution according to ℓ1-norm. So we first draw the unit circle of the ℓ1-norm, and gradually enlarge this circle, the first point touches the red line is the optimal point. And normally and mostly, the point is one of the vertices of the circle. This is what we mean by sparsity because one of the coordinate is 0. However, as you can see from the black line case, if the object function has a slope which is parallel to one of the edge of the unit circle, the sparsity effect may not be realized. That is why the correct way to phrase this is that ℓ1-norm encourages sparsity, but it does not guarantee sparsity. In fact, any ℓp-norm, with $0 \leq p < 1$ potentiality gives better sparsity, see figure 3, however, they are less deployed due to non-convexity.

Back to our problem, after transforming the surface, we enforce normals to align to global coordinate axes. This is defined by the following Cubeness term,

$$CUBENESS := \sum_{i \in V} \|R_i \hat{n}_i\|_1$$

V stands for vertices, R_i is the rotation matrix for a vertex, and $\hat{n}_i$ is the unit area-weighted norm of the i-th vertex. The last parameter is calculated in steps. First find all the normals from the surrounding triangles, then calculate the areas of those triangles, the weights are proportional to the areas, finally the interpolation of all weighted normals can be seen as the normal vector of the center vertex, see figure 4.

However, this is not enough to model the problem. Only a cubeness term can lead fine designed meshes to a pure cube, so we need to find a way to preserve the sophisticated details. Therefore,

the intuition should be: our deformed mesh should also be similar to (maintain the same content of) the original one.

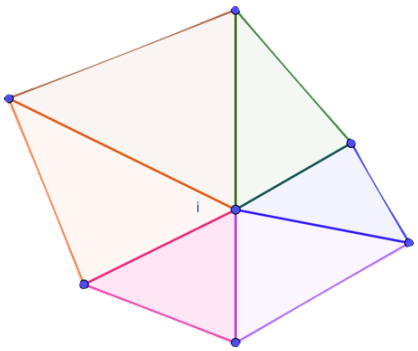

Figure 4: Areas of one-ring triangles

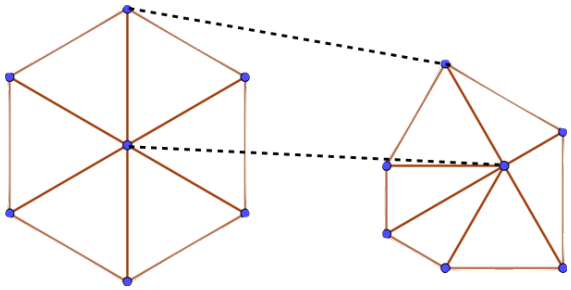

Figure 5: ARAP defined with "spokes and rims" edges

2.1.2 As-Rigid-As-Possible

As-Rigid-As-Possible Surface Modeling [6] is a method to address this issue. This is a work from Sorkine and Alexa in year 2007, and it's cited and used by lots of different papers.

$$ARAP := \sum_{i \in V} \sum_{j \in N(i)} \left\| R_i d_{i,j} - \tilde{d}_{i,j} \right\|_F^2 \tag{1}$$

Here, the formula 1 enforces the edges of deformed state $\tilde{d}_{i,j}$ is identical to the original state $d_{i,j}$ up to a rigid transformation (here means rotation R_i). $N(i)$ is the "spokes and rims" edges, see figure 5.

Once we have defined the cubeness term and ARAP term, the objective function is within reach.

$$\min_{\tilde{V}, R_i} = \sum_{i \in V} \sum_{j \in N(i)} \frac{w_{i,j}}{2} \left\| R_i d_{i,j} - \tilde{d}_{i,j} \right\|_F^2 + \lambda a_i \left\| R_i \hat{n}_i \right\|_1 \tag{2}$$

As all kinds of optimization problems, when putting two different terms together, usually additional parameters are added in front of at least one of them to control the weights, and each term is emphasized by the value of the weights. The authors [4] add three, $w_{i,j}$ in front of ARAP term, and λ and a in front of cubeness term. $w_{i,j}$ represents the cotangent weight, it is the sum of cotangent of angles opposite of edge (i, j), the edge shared by two triangles, see in figure 6. a is barycentric area, the area of the region bound by barycenters of neighboring triangles, figure 7. λ is a totally free parameter for user to control the deformation. More will be discussed in section 3.

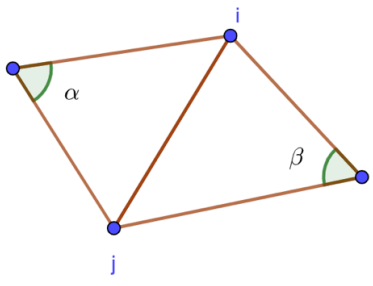

Figure 6: Cotangent weight defined around the edge (i, j)

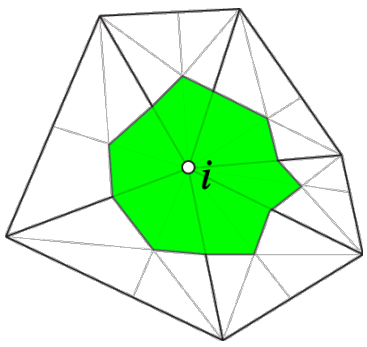

Figure 7: Barycentric area at vertex i

Specifically, cotangent weights are used for mitigating the problem caused by meshing bias during deformation [6]. Barycentric area enforces large cubeness on vertices surrounded by large triangles, and let small triangles to preserve the local detail.

2.2 Optimization

Once we get the objective function, the next question to answer is how to solve it. The Cubic Stylization paper follows the ARAP paper, and uses the local-global scheme. They proposed that first calculate the rigid body transformation locally, then update the vertices in a global manner. As the rigid body transformation only holds locally. If it holds for entire mesh, then the deformation will collapse to a rotation which has no stylization effects.

Let's have a look at the local step, local in the sense of "spokes and rims" region. Firstly, the objective is re-written as follows.

$$R_i^\star = \underset{R_i \in SO(3)}{\arg\min} \frac{W_i}{2} \left\| R_i D_i - \tilde{D}_i \right\|_F^2 + \lambda a_i \left\| R_i \hat{n}_i \right\|_1 \tag{3}$$

Then the author reformulate the function:

$$\underset{z, R_i \in SO(3)}{minimize} \quad \frac{W_i}{2} \left\| R_i D_i - \tilde{D}_i \right\|_F^2 + \lambda a_i \left\| z \right\|_1 \qquad s.t. \quad z - R_i \hat{n}_i = 0 \tag{4}$$

The advantage of doing this is that they can use the method call Alternating Direction Methods of Multipliers.

2.2.1 Alternating Direction Method of Multipliers

Alternating Direction Method of Multipliers, or abbreviated as ADMM [1], is an algorithm that solves convex optimization problems by breaking them into smaller pieces, each of which are then easier to handle. It is a variant of the standard augmented Lagrangian method. The later is formulated as minimizing a function with constraints,

$$\min f(x) \quad s.t. \quad c_{i \in I}(x) = 0.$$

The update rule for k-th iteration is

$$\begin{aligned} \min \Phi_k(x) &= \min f(x) + \sum_{i \in I} \beta_i c_i(x) + \frac{\rho_k}{2} \sum_{i \in I} c_i(x)^2 \\ \beta_{k+1} &\leftarrow \beta_k + \rho_k c_i(x_k) \end{aligned} \tag{5}$$

ADMM alters the problem with a trick. If the object function is linear separable, then it replace the second chuck with a different parameter, and add a constraint like equation 4.

$$\min f(x) + g(x) \iff \min f(x) + g(y) \quad s.t. \quad x = y.$$

Following update rules 5, the update rule for ADMM can be derived as

$$\begin{aligned} \min \Phi_k(x) &= \min f(x) + g(y) + \beta(x - y) + \frac{\rho_k}{2}(x - y)^2 \\ &= \min f(x) + \beta x + \frac{\rho_k}{2}(x - y)^2 \\ &= \min f(x) + \frac{\rho_k}{2}(x - y + u)^2 + const, \\ \min \Phi_k(y) &= \min f(x) + g(y) + \beta(x - y) + \frac{\rho_k}{2}(x - y)^2 \\ &= \min g(y) + \frac{\rho_k}{2}(x_{k+1} - y + u)^2 + const, \\ u_{k+1} &\leftarrow u_k + x_{k+1} - y_{k+1} \end{aligned} \tag{6}$$

Given our optimization problem 3, the update rules can be derived by substituting equations 6 piece by piece

$$\begin{aligned} R_i^k &\leftarrow \underset{R_i \in SO(3)}{\arg\min} \frac{W_i}{2} \left\| R_i D_i - \tilde{D}_i \right\|_F^2 + \frac{\rho^k}{2} \left\| R_i \tilde{n}_i - z + u^k \right\|_2^2 \\ z^{k+1} &\leftarrow \underset{z}{\arg\min} \lambda a_i \|z\|_1 + \frac{\rho^k}{2} \left\| R_i^{k+1} \tilde{n}_i - z + u^k \right\|_2^2 \\ \tilde{u}^{k+1} &\leftarrow u^k + R_i^{k+1} \tilde{n}_i - z^{k+1} \\ \rho^{k+1}, u^{k+1} &\leftarrow update(\rho^k) \end{aligned} \tag{7}$$

The step of updating R is called orthogonal Procrustes [3]; z is calculated as LASSO problem using shrinkage step [1].

After run the ADMM for solving the optimization problem, we finally get R for every vertex. As we discussed in the beginning of this section, we need a global step to update the positions of vertices. This is done by find the solution of a linear equation system.

$$\sum_{j \in N(i)} w_{i,j} \tilde{d}_{i,j} = \sum_{j \in N(i)} \frac{w_{i,j}}{2} (R_i + R_j) d_{i.j} \tag{8}$$

We want the positions of edge of deformed state $\tilde{d}_{i,j}$ to agree on all neighboors' rigid body transformations calculated in local step to an extent.

2.2.2 Affine Progressive Mesh

As sometimes the sophisticated meshes contains millions of face, see figure 9, thus, the author also adapts a method called Affine Progressive Mesh into the process. This is a work published by Manson and Schafer in year 2011 [5]. It aims to address the runtime problem on high resolution meshes.

The core idea is to downsample the total vertices, take note of sequence of vertices merges, after some other time consuming operations, then expand them back by fitting best affine transformation, shown in figure 8. For high degree of cubeness, the local rigidity is hard to hold. So we loose the constraint by substituting the rigid transformation to an affine transformation.

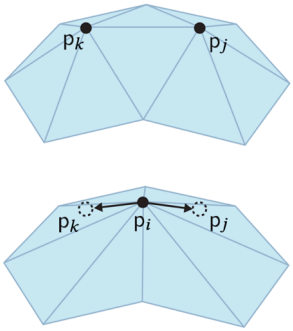

Figure 8: Edge collapses

Figure 9: Numbers of faces of different meshes

Note that this worked as a pre-processing step before cubic stylization, which means the cubic stylization runs on the coarsest mesh after edge collapsing. After deformation, the vertices are split again to visualize the final result on the original resolution. The reason to emphasis this is that according to the authors, the runtime is measured only for cubic stylization, not the whole downsampling to upsampling steps, which is typically several times longer than the stylization.

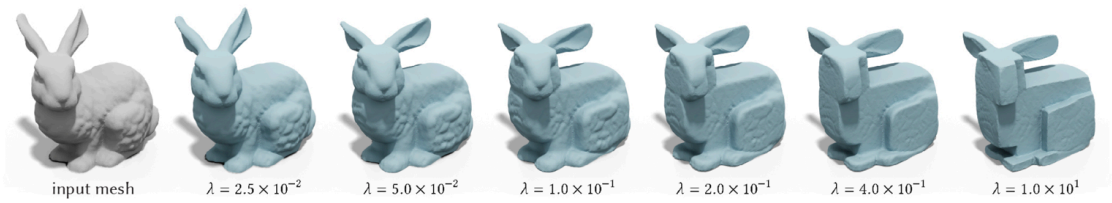

Figure 10: Control the cubeness through changing the value of λ

3 Artistic Controls

Given the objective function 2 let's have a look of what kind of artistic controls are provided by the model. As λ is the parameters before the cubeness term, therefore, a large lambda stands for higher emphasis on cubeness, see in figure 10. We can also split the single λ into a set of parameters. For example, 3, and each of them takes care of the cubeness in one direction of the global coordinate. Check the illustration on the left part of the figure 11. This can go further by generating a gauss map with continuous changing value for λ, and mapped onto the mesh. Different parts have different degrees of cubeness. Moreover, a modeler can insert several local

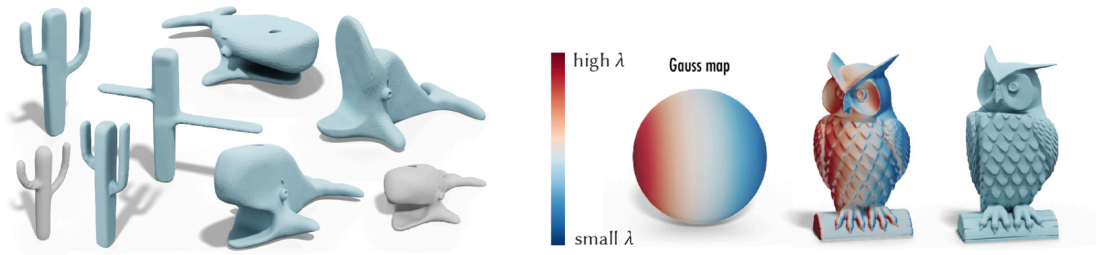

Figure 11: Control the cubeness through changing the value of λ_d (for $d \in \{1, 2, ..., n\}$)

coordinate systems into the $\ell 1$-norm, and create a polyhedric stylization, no matter uniform or non-uniform cube orientations, see figure 12.

Figure 12: Apply a coordinate transformation inside $\ell 1$-norm to generalize cubic stylization to polyhedrons.

4 Summary

In general, this is an interesting work which draws a link between art and sciences. It creates a clear stated mathematical tool for cubic stylization while provides some free controls at the same time. The mainly downside still remains at computational time, which barely reaches on-the-fly

performance with a commodity hardware setup. Nevertheless, this work shoots a beam of light to model more intricate styles.

References

[1] Stephen Boyd, Neal Parikh, Eric Chu, Borja Peleato, and Jonathan Eckstein. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Found. Trends Mach. Learn.*, 3(1):1–122, January 2011.

[2] Hsieh-Chung Chen. Gradient descent for optimization problems with sparse solutions. *Doctoral dissertation, Harvard University, Graduate School of Arts & Sciences*, 2016.

[3] John C. Gower. Procrustes methods. *WIREs Computational Statistics*, 2(4):503–508, 2010.

[4] Hsueh-Ti Derek Liu and Alec Jacobson. Cubic stylization. *ACM Transactions on Graphics*, 2019.

[5] Josiah Manson and Scott Schaefer. Hierarchical deformation of locally rigid meshes. *Computer Graphics Forum*, 30(8):2387–2396, 2011.

[6] Olga Sorkine and Marc Alexa. As-Rigid-As-Possible Surface Modeling. In Alexander Belyaev and Michael Garland, editors, *Geometry Processing*. The Eurographics Association, 2007.