DOI: 10.1111/cgf.14355
Eurographics Symposium on Geometry Processing 2021
K. Crane and J. Digne
(Guest Editors)

Volume 40 (2021), Number 5

Gauss Stylization: Interactive Artistic Mesh Modeling based on Preferred Surface Normals

M. Kohlbrenner[†], U. Finnendahl[†], T. Djuren, M. Alexa

TU Berlin, Computer Graphics Group

Abstract

Extending the ARAP energy with a term that depends on the face normal, energy minimization becomes an effective stylization tool for shapes represented as meshes. Our approach generalizes the possibilities of Cubic Stylization: the set of preferred normals can be chosen arbitrarily from the Gauss sphere, including semi-discrete sets to model preference for cylinder- or cone-like shapes. The optimization is designed to retain, similar to ARAP, the constant linear system in the global optimization. This leads to convergence behavior that enables interactive control over the parameters of the optimization. We provide various examples demonstrating the simplicity and versatility of the approach.

Keywords: geometry processing, geometric stylization, shape modeling

CCS Concepts

• **Computing methodologies** → **Mesh models; Mesh geometry models;** Non-photorealistic rendering;

1. Introduction

By the *style* of a shape we refer to properties of the geometry that are relevant for its aesthetics or distinctive appearance. *Stylization* aims to perform directed alterations in order to establish a certain style. In computer graphics, automated stylization techniques have a long tradition especially in rendering, where a given shape is turned into an image following different visual principles [Str02, GG01]. The stylization of the shape itself is much less explored. It has recently garnered interest with direct digital manufacturing ("3D printing") becoming affordable [BCMP18].

Stylization of a shape can be performed at different scales, from the change or transfer of local details (e.g. through differential coordinates and local coordinate systems [TSS*11, SS10, SGW06, SBS07, WDB*07, BH11, HFAT07]) to the manipulation of more global shape-related characteristics. We argue that styles at large scale are characterized by the *Gauss map* of the shape, that is, the distribution of its surface normals [Koe90]. One may argue that the generation of bas-reliefs from shape is an instance of this style, for which several automatic approaches exist [SBS07, WDB*07, BH11], even lifted to truly three-dimensional surfaces [SPSH14].

Liu and Jacobson [LJ19] developed a stylization technique that 'cubifies' a shape. The method is based on the addition of an ℓ_1 regularization term to the widely used As-rigid-as-possible (ARAP) surface modeling approach [SA07]. The regularization encourages alignment of the rotated vertex normals with the coordinate axes.

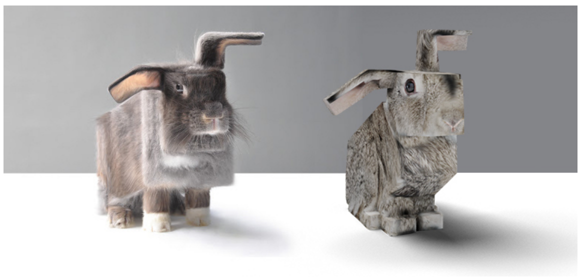
Figure 1: *The artistic ©Anicube Bunny (left) by Aditya Aryanto is based on severely restricting the surface normals, but not just to those of a cube. Gauss stylization allows creating a similar style (right). (Used with permission. ©Bunny texture (right) by yellokab under CC BY.)*

We briefly review this approach in Section 2. The resulting visual effect has sparked enormous interest, highlighting the need for simple yet expressive stylization tools.

Coupling the control over the surface normals to a p-norm limits the choice of preferred surface normals. Generalized metrics still require the normals to be point symmetric, i.e. if $\mathbf{n}$ is a normal among the preferred directions then so is $-\mathbf{n}$. This may appear a minor limitation, but note that already the simplest of all polyhedra, the tetrahedron, lacks this central symmetry. More importantly, many artistic styles deviate from symmetric normals (see, for example, Figure 1).

† equal contribution

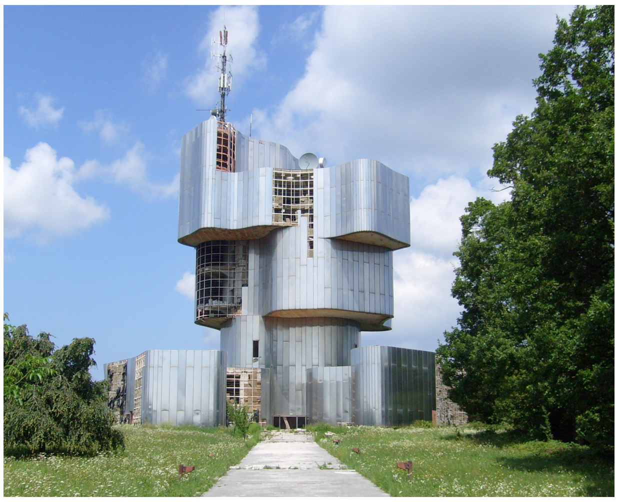

Figure 2: *Monument to the uprising of the people of Kordun and Banija by Vojin Bakić. A style that our method can imitate. © Picture by Sandor Bordas under under CC BY-SA.*

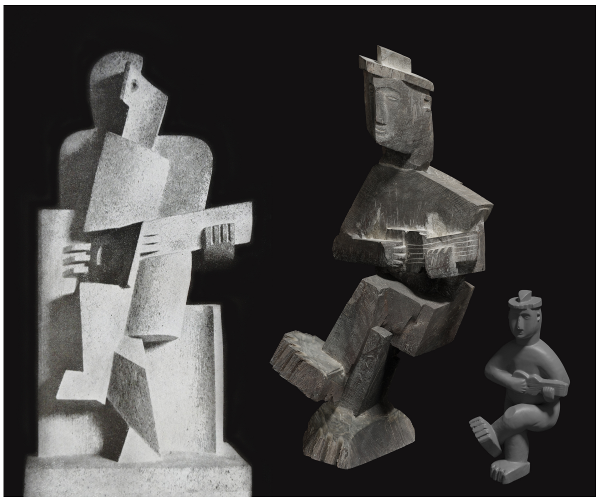

Figure 3: *"Seated Man with a Guitar" cubism sculpture by Jacques Lipchitz (left). Our interactive tool allows mimicking the style by preferring different normals in certain regions of the model (right). Original model © Sculpture Man with Guitar by Världskulturmuseerna under CC BY.*

In this work, we introduce an interactive modeling technique based on flexible control over the surface normals of a shape. The freedom of choosing arbitrary normal direction directly addresses the need for intuitive modeling tools that can be used by casual users. Direct feedback to user interaction is crucial for predictable editing of a shape. Our main technical contribution provides the necessary framework for such an approach: a formulation of a functional together with its optimization that is tailored to provide precise yet interactive control over the surface normals.

This is achieved by inserting a penalty term to the ARAP modeling framework, whose optimization can be performed locally despite the global non-linear nature – see Section 3 for details. Unlike other approaches, the penalty term is applied to the face normals, providing more precise control. Moreover, our general formulation naturally extends to sets of surface normals that are not discrete sets on the Gauss sphere. This enables styles that would prefer primitives such as cylinders or cones and may be used to generate the characteristics of the monument Petrova Gora as shown in Figure 2 or surfaces that curve only in one direction such as, notably, cloth.

Our approach retains the favorable properties of minimizing the ARAP energy, namely, that the global step is linear and the system matrix remains constant throughout the minimization. This not only enables fluid interaction in an intuitive graphical interface, providing controls that are simple to use for experts and novices. In addition, the unchanged system matrix also makes it easy to combine the stylization with other, more established, modeling controls such as point constraints (see Section 3.3).

We provide details on the implementation of the optimization as well as an exploration of the parameters in Section 4. Results, shown throughout the paper, demonstrate that our approach enables a wide variety of styles, in particular inspired by cubism, but not limited to the surface normals of a cube (Figure 3). The results are discussed and compared to cubic stylization in Section 5.

2. Background and Motivation

The main idea for our stylization approach is that the desired geometry at a large scale is characterized by a given Gauss image. Given an input shape, we want to preserve the characteristic details of the geometry, but deform the shape at a larger scale so that its surface normals are close to the preferred normals.

As a general modeling framework we rely on the idea of As-rigid-as-possible (ARAP) surface modeling [SA07]. Then, similar to Cubic Stylization [LJ19], we add a term that penalizes the use of undesirable surface models. In the following, we briefly recall the main ideas of these methods.

The modification of the mesh will only change the original vertex positions $\hat{\mathbf{v}}_i \in \hat{\mathbf{V}}$, while the combinatorics of the mesh is represented as a set of vertices $\mathcal{V}$, edges $(i, j) \in \mathcal{E}$ and triangular faces $(i, j, k) \in \mathcal{F}$ remains constant. We will denote the changed vertex positions as $\mathbf{v}_i \in \mathbf{V}$.

2.1. Related work

Liu and Jacobson [LJ19] provide an excellent overview of mesh modeling tools for aesthetic modeling and generating specifiable styles. We focus here on work not covered in their overview. Several approaches in the field of voxel/lego art [LYH*15, TSP13] and polycubes [THCM04, HJS*14] lead to 'cube stylized' results. Those methods are not suitable to preserve the local structure of the input [LJ19]. Alexa [Ale21] investigates the error-controlled approximation of geometry with a fixed set of normals, yet the approach is not geared towards interactive stylization. Stein et al. [SGC18] transform a given input mesh to a stylized version comprised of developable surface patches. Similar to our approach, they link the desired style to a specific Gauss image consisting of a network of curves.

While not introduced as stylization methods, filtering techniques could be used for stylization and some of them are related in their technical approach. Some filtering techniques mainly apply the filter to the face normals and then fit the geometry to the filtered normals [WFL*15, ZDH*19]. The mesh denoising method by He and Schaefer [HS13] is based on ℓ_0 minimization, similar to Cubic Stylization, and uses auxiliary variables similar to our optimization approach. The basic idea of ARAP has been generalized to arbitrary local non-linear objectives [BDS*12] minimized by projection. It is quite likely that 'style' could be encoded projection operator and is a worthwhile topic for future research.

2.2. ARAP Surface Modeling

The idea of ARAP surface modeling [SA07] is to preserve the local shape by penalizing locally non-rigid transformations. This penalty is minimized while respecting modeling constraints, typically given as some fixed vertex positions.

Let $\hat{\mathbf{e}}_{ij} = \hat{\mathbf{v}}_j - \hat{\mathbf{v}}_i, \mathbf{e}_{ij} = \mathbf{v}_j - \mathbf{v}_i$ be the edge vectors of the original geometry $\hat{\mathbf{V}}$ and modified geometry $\mathbf{V}$. Consider an approximation of the local rigid transformation $\mathbf{R}_i$ in a local neighborhood $N(i)$ of vertex i. Then we measure the squared distance of rotated original edges from the modified ones:

$$E_i(\mathbf{V},\mathbf{R}_i) = \sum_{j\in N(i)} \frac{w_{ij}}{2}\|\mathbf{e}_{ij} - \mathbf{R}_i\hat{\mathbf{e}}_{ij}\|^2. \tag{1}$$

The local ARAP energy for vertex i and modified geometry $\mathbf{V}$ is obtained as the minimizer among all rotations $\mathbf{R}_i$, i.e. as $\min_{\mathbf{R}_i} E_i$. Summing up over all vertices results in the global ARAP energy:

$$E_{\text{ARAP}}(\mathbf{V},\{\mathbf{R}_i\}) = \sum_{i\in\mathcal{V}} E_i(\mathbf{V},\mathbf{R}_i) \tag{2}$$

The appeal of this energy is that it can be efficiently minimized by block-coordinate descent, typically called *local-global* optimization. For fixed updated vertex positions $\mathbf{V}$ computing optimal rotations $\{\mathbf{R}_i\}$ is local, i.e. can be done by only considering the neighborhood $N(i)$. The resulting problem can be solved by the polar decomposition of a weighted cross co-variance matrix of the edge vectors. On the other hand, optimizing the vertex positions $\mathbf{V}$ for fixed rotations is a standard linear least squares problem. The system matrix turns out to be the Laplacian matrix using w_{ij} as the edge weights. A particular feature of this optimization is that the change of the rotation has no effect on the system matrix. This means it can be factored in a pre-process and then the global step only amounts to back- and forward substitution.

It is common to use the *cotan* Laplacian for ARAP surface modeling, so the w_{ij} are the cotangent weights of the original edge $\hat{\mathbf{e}}_{ij}$ [PP93, MDSB03]. For the neighborhoods $N(i)$ Chao et al. [CPSS10] have shown that it is beneficial to use all half-edges of triangles incident to i, rather than just the edges incident on the vertex.

2.3. Cubic Stylization

The idea of Cubic stylization [LJ19] is to "encourage" the surface normal of a shape to align with the main axes of the coordinate system, while preserving the overall structure and detail of a given shape. The latter goal is modeled as minimizing the ARAP energy. The surface normals of the deformed geometry are approximated in the vertices by applying the optimal rotation to the original vertex normals $\{\hat{\mathbf{n}}_i\}$. The energy term considered for these rotated normals is the ℓ_1 norm, which is smallest for the canonical coordinate directions. The resulting energy is:

$$E_{\square}(\mathbf{V},\{\mathbf{R}\}) = E_{\text{ARAP}}(\mathbf{V},\{\mathbf{R}_i\}) + \lambda \sum_{i\in\mathcal{V}} a_i\|\mathbf{R}_i\hat{\mathbf{n}}_i\|_1, \tag{3}$$

where a_i is the barycentric area associated to vertex i. The parameter λ allows balancing between the constraint on the normals and preserving the original geometry. For the optimization of this energy it is convenient that the global step is unaltered. Optimizing $\mathbf{R}_i$ remains local but becomes more difficult. An efficient ADMM scheme is suggested for this step.

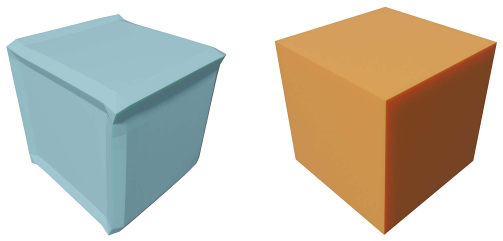

Figure 4: *Cubic stylization of a cube. A penalty function based on vertex normals leads to unexpected transformations on sharp edges (left). Gauss stylization is based on face normals and preserves shapes that already conform to the desired style.*

We note that $\mathbf{R}_i\hat{\mathbf{n}}_i$ is an approximation of the normals in the deformed state as (1) using vertex normals leads to an integration of the surface normals around vertex i and (2) the rotated normal of the original geometry is not the same as the vertex normal of the modified geometry. The first issue leads to discretization artifacts around sharp edges and corners. Figure 4 shows the 'cubification of a cube', demonstrating how Cubic stylization modifies the geometry of a perfect cube around the edges because the vertex normals are not aligned with those of a cube.

The choice of the ℓ_1 norm has been motivated by its use in sparse approximation: the non-vanishing (and discontinuous) gradient in the extrema helps generating solutions that are exactly zero in many components. In our early experiments we have found that for the purpose of stylization with given normals these properties of a sparsity inducing norm are probably not necessary. In particular, maximizing the ℓ_2 norm of weighted inner products of the normals with a discrete set of preferred directions also leads to 'cubified' results. While the style is slightly different, it would be difficult to objectively prefer one result over the other (Figure 5).

3. Method

Similar to cubic stylization, we add a term to the original ARAP energy that penalizes deviation of the surface normals from preferred normal directions. In our formulation, the preferred normal directions are modeled by a preference function $g : \mathbb{S}^2 \mapsto \mathbb{R}$, where

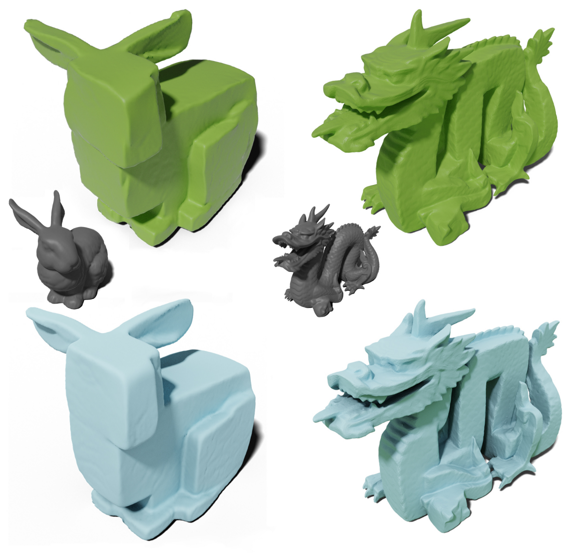

Figure 5: *Cubic Stylization using the maximum of ℓ_2 weighted scalar products with preferred normals (top row) and Cubic Stylization by Liu and Jacobson [LJ19] (bottom row).*

higher values indicate higher preference. Elements of $\mathbb{S}^2$ will be represented as unit vectors $\mathbf{x} \in \mathbb{R}^3, \mathbf{x}^\mathsf{T}\mathbf{x} = 1$. This function would be ideally applied to the true surface normals of the modified geometry $\mathbf{V}$, i.e. our aim is to minimize:

$$E_G(\mathbf{V}, \{\mathbf{R}_i\}) = E_{\text{ARAP}}(\mathbf{V}, \{\mathbf{R}_i\}) - \mu \sum_{f \in \mathcal{F}} g(\mathbf{n}_f), \tag{4}$$

where $\mathbf{n}_f$ is the unit normal vector associated to face: f, i.e. satisfies

$$\mathbf{n}_f^\mathsf{T}\mathbf{e}_{ij} = 0, \qquad (i,j) \in f. \tag{5}$$

and the parameter μ weighs the preference for the desired normals.

Regardless of the particular choice of g, however, this function will be difficult to minimize. In particular, the global part is no longer a linear system with constant system matrix. Our idea for an approximation that admits minimization in an interactive framework is inspired by ARAP: notice that the rigidity term in ARAP provides 'desired' edge vectors $\frac{1}{2}(\mathbf{R}_i + \mathbf{R}_j)\mathbf{e}_{ij}$, whose optimization is performed in the local step (implicitly, by optimizing $\mathbf{R}_i$ and $\mathbf{R}_j$). We likewise, introduce desired edges $\mathbf{e}^*_{ij}$, reflecting the maximization of g. This idea leads to the following setup:

$$\begin{aligned} E^*_G(\mathbf{V}, \mathbf{R}, \mathbf{n}^*, \mathbf{e}^*) &= E_{\text{ARAP}}(\mathbf{V}, \{\mathbf{R}_i\}) - \mu \sum_{f \in \mathcal{F}} g(\mathbf{n}^*_f) \\ &+ \lambda \sum_{(i,j)\in\mathcal{E}} \frac{w_{ij}}{2} \|\mathbf{e}_{ij} - \mathbf{e}^*_{ij}\|^2, \qquad (6) \\ &\text{s.t. } \mathbf{n}^{*\mathsf{T}}_f \mathbf{e}^*_{ij} = 0, \quad (i,j) \in f \end{aligned}$$

Here we have replaced the face normal $\mathbf{n}_f$ by the variable $\mathbf{n}^*_f$, which is constrained to be orthogonal to the desired edges $\mathbf{e}^*_{ij}$.

In this setup, the vertex positions $\mathbf{V}$ can still be computed as the solution of a linear system with fixed system matrix. The optimization of all other terms is non-linear. Below we describe how to compute a descent step in a local fashion. We first explain our design of the preference function g and then how to perform the optimization steps.

3.1. Preference Function

We suggest to model g as a mixture of Gaussians, which often appears in the context of geometric modeling as Blobs [Bli82]. Let $\mathbf{n}, \|\mathbf{n}\| = 1$ be a preferred normal direction. Then a Gaussian centered around $\mathbf{n}$ is given by:

$$\exp\left(-\frac{1}{2}\frac{\|\mathbf{x}-\mathbf{n}\|_2^2}{\nu}\right) = \exp\left(-\frac{1}{2}\frac{\|\mathbf{x}\|_2^2 + \|\mathbf{n}\|_2^2 - 2\mathbf{x}^\mathsf{T}\mathbf{n}}{\nu}\right). \tag{7}$$

Restricting $\mathbf{x}$ to the unit sphere, we observe that the squared lengths of $\mathbf{x}$ and $\mathbf{n}$ are both one, and this expression turns into:

$$\exp\left(\sigma \mathbf{x}^\mathsf{T}\mathbf{n}\right) \tag{8}$$

for some positive parameter σ. Up to a constant factor, this is known as the von Mises-Fisher distribution [Wat82].

We want to stress that the construction based on the von Mises-Fisher distribution may look like forming a probability distribution, but is meant to be only expressing preference. The main feature of Gaussians we exploit is that they can be added, and that the maxima are generally close to the means of the individual Gaussians. In addition, the computation of values and derivatives is simple and fast, enabling interactive editing of the preference function by defining desired normals, with the stylization changing in real time (see accompanying video). In the following, we discuss the details for the discrete case as well as for a restricted class of semi-discrete sets. Figure 6 shows several examples of preference functions constructed with this method.

Note that while we use the names of canonical shapes such as 'cube' or 'cylinder', when describing the surface normals and corresponding g functions, the surfaces that have similar Gauss images may be very far from these primitives. For example, we describe the monument shown in Figure 2 as having 'cylinder' normals.

Discrete Normals Using the von Mises-Fisher distribution, we can model a preference function for several preferred normal directions as a simple linear combination:

$$g(\mathbf{x}) = \sum_k w_k \exp(\sigma \mathbf{x}^\mathsf{T}\mathbf{n}_k) = (\exp(\sigma \mathbf{x}^\mathsf{T}\mathbf{n}_0), \ldots)\mathbf{w}. \tag{9}$$

The weights $\mathbf{w}$ provide control over the relative influence of the normals. Intuitively, and backed up by practical evidence, it is preferable that g takes on the same value for all preferred normals. This requires solving a linear system arising from the conditions $g(\mathbf{n}_k) = 1$:

$$\begin{pmatrix} \exp(\sigma) & \exp(\sigma\mathbf{n}_0^\mathsf{T}\mathbf{n}_1) & \exp(\sigma\mathbf{n}_0^\mathsf{T}\mathbf{n}_2) & \cdots \\ \exp(\sigma\mathbf{n}_1^\mathsf{T}\mathbf{n}_0) & \exp(\sigma) & \exp(\sigma\mathbf{n}_1^\mathsf{T}\mathbf{n}_2) & \cdots \\ \vdots & \vdots & \vdots & \ddots \end{pmatrix} \mathbf{w} = \mathbf{1} \tag{10}$$

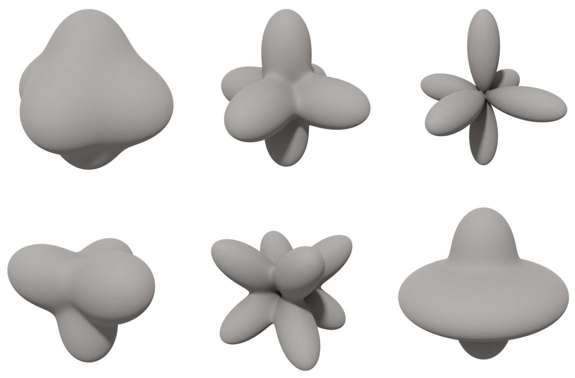

Figure 6: *Visualization of different functions g with the unit sphere as input. Uniform weights were used for the normals of a cube with* $\sigma = 2$, 5 *and* 10 *(top row), a tetrahedron, an octahedron and a cylinder (bottom row, left to right).*

Apart from taking on the same value in the preferred directions, it is also important that g has local maxima in or close to $\mathbf{n}_k$. This may not be the case and require the increase of σ.

The *unconstrained* gradient of g is:

$$\nabla_g(\mathbf{x}) = \sigma \sum_k w_k \exp(\sigma \mathbf{x}^\mathsf{T} \mathbf{n}_k)\, \mathbf{n}_k. \tag{11}$$

For the constraint $\mathbf{x}^T\mathbf{x} = 1$, a critical point is attained if $\nabla_g(\mathbf{x})$ and $\mathbf{x}$ are parallel. Ideally, the local maxima of g coincide with the $\{\mathbf{n}_i\}$. This does happen if the preferred normals exhibit certain symmetries. Otherwise increasing σ moves the maxima closer to the $\mathbf{n}_i$, because it reduces the effect of the term involving $\mathbf{n}_i$ on other normal directions. We prefer the in practice small deviation of the maxima from the $\mathbf{n}_i$ because it avoids a global optimization of g, enabling interactive modelling.

For use in Newton optimization, we note that the unconstrained Hessian is:

$$\mathbf{H}_g(\mathbf{x}) = \sigma^2 \sum_k w_k \exp(\sigma \mathbf{x}^\mathsf{T} \mathbf{n}_k)\, \mathbf{n}_k \mathbf{n}_k^\mathsf{T}. \tag{12}$$

Semi-Discrete Normals In principle, g can be generalized to arbitrary sets by appropriate integration. We specifically consider the Gauss images of cylinders and cones, which form circles on the Gauss sphere. Let $\mathbf{a}$ be the axis of the cylinder or cone, then the normals satisfy the condition $\mathbf{x}^\mathsf{T}\mathbf{a} = d$, where d specifies the opening angle of the cone ($d = 0$ specifies a cylinder). To avoid discontinuities we suggest the term

$$c(\mathbf{x}) = \exp\left(\sigma\left(1 - (\mathbf{x}^\mathsf{T}\mathbf{a} - d)^2\right)\right), \tag{13}$$

with gradient

$$\nabla_c(\mathbf{x}) = -2\sigma(\mathbf{x}^\mathsf{T}\mathbf{a} - d)c(\mathbf{x})\, \mathbf{a} \tag{14}$$

and Hessian

$$\mathbf{H}_c(\mathbf{x}) = 2\sigma\left(2\sigma(\mathbf{x}^\mathsf{T}\mathbf{a} - d)^2 - 1\right)c(\mathbf{x})\, \mathbf{a}\mathbf{a}^\mathsf{T} \tag{15}$$

for the preference function.

In practice, it may be useful to add the discrete normals term for the two normals of the axis. One may think of this intuitively as adding the normals of the bases of the cone or cylinder. For these discrete normals we use the term introduced previously and balance the terms using a user controllable weight. More circles and discrete normals could be added, but in our experiments we have restricted circles to a single fixed axis. This avoids problems with asymmetry in g, which would require global optimization. A relevant practical case that can be modeled in this setup are the normals of a double cone. The symmetry in this scenario allows solving for $g = 1$ in the constraints similar to the discrete case.

3.2. Minimization: Auxiliary Variables / Local Step

For the following discussion we assume that the edge vectors $\mathbf{e}_{ij}$ are fixed. The optimal choice of $\mathbf{R}_i$ is unaltered compared to the original ARAP formulation. They can still be computed locally using e.g. the polar decomposition.

Optimizing $\mathbf{e}^*_{ij}$ and $\mathbf{n}^*_f$ for fixed $\mathbf{e}_{ij}$ is, unlike the rotations, a global problem, because they are coupled through the orthogonality constraint – and the faces are connected across edges. To avoid global non-linear optimization, we use ADMM updates on an augmented Lagrangian that can be calculated locally:

$$L = E_{\mathrm{ARAP}}(\mathbf{V}, \{\mathbf{R}_i\}) - \mu \sum_{f \in \mathcal{F}} g(\mathbf{n}^*_f) + \lambda \sum_{(i,j) \in \mathcal{E}} \frac{w_{ij}}{2} \|\mathbf{e}_{ij} - \mathbf{e}^*_{ij}\|^2 + \sum_{f \in \mathcal{F}} \sum_{(i,j) \in f} \left(\rho u_{fij} \mathbf{n}^{*\mathsf{T}}_f \mathbf{e}^*_{ij} + \frac{\rho w_{ij}}{2}\left(\mathbf{n}^{*\mathsf{T}}_f \mathbf{e}^*_{ij}\right)^2\right). \tag{16}$$

We are using the scaled ADMM approach [BPC*11, Section 3.1.1] with the scaled dual variable u_{fij}. The factor ρ for the quadratic regularizer is weighted by w_{ij} similar to the quadratic coupling term and the weights used in ARAP. This choice weights simplifies the computation of the global step and, in particular, leads to fixed system matrix. Notice that the constraint is not linear but *bi-affine*. This, in fact, slightly simplifies the ensuing optimization using ADMM [BPC*11]. For the parameters controlling the regularizer we choose $\rho = \mu\lambda$.

In the following we explain our initialization and then detail the individual update steps for the variables following the ADMM approach for our setting [BPC*11, Section 9.2].

Initialization For initialization, we set $\mathbf{n}^*_f$ to the face normals of the geometry given by $\mathbf{e}_{ij}$. Based on this choice, a natural initialization for the edges is $\mathbf{e}^*_{ij} = \mathbf{e}_{ij}$, because this zeroes all terms that involve $\mathbf{e}^*_{ij}$.

We reinitialize the ADMM variables in each local step since the geometry can change significantly in the global step. Using the values from the previous iteration may lead to instabilities because of the discrepancies between the updated geometry and the locally optimized edges and normals. Consequently, we initialize the scaled variables as $u_{fij} = 0$.

Normals For updating the normals, the ADMM approach leads to the following optimization problem:

$$\mathbf{n}_f^* \leftarrow \arg\min_{\mathbf{n}_f^*} L_n \tag{17}$$

$$L_n = -g(\mathbf{n}_f^*) + \lambda \sum_{(i,j)\in f} \frac{w_{ij}}{2} \left(u_{fij} + \mathbf{n}_f^{*\top} \mathbf{e}_{ij}^*\right)^2 \tag{18}$$

The ARAP energy is independent of the normals and has no effect on the Lagrangian. Notice that the problem is local, i.e. only involves one surface normal and the variables connected to the edges of this face. Our choice of ρ involving μ leads to the local problem becoming independent of μ. The gradient w.r.t. the current normal is given by:

$$\nabla_{L_n}(\mathbf{n}_f^*) = -\nabla_g(\mathbf{n}_f^*) + \lambda \sum_{(i,j)\in f} w_{ij} \left(\mathbf{e}_{ij}^{*\top} \mathbf{n}_f^* + u_{fij}\right) \mathbf{e}_{ij}^* \tag{19}$$

For the Hessian, the additional contribution of the quadratic term coming from the regularizer is:

$$\mathbf{H}_r(\mathbf{n}_f) = \lambda \sum_{(i,j)\in f} w_{ij} \mathbf{e}_{ij}^* \mathbf{e}_{ij}^{*\top}. \tag{20}$$

The unconstrained Newton step for updating the normal then amounts to solving:

$$\left(-\mathbf{H}_g(\mathbf{n}_f^*) + \mathbf{H}_r(\mathbf{n}_f^*)\right) \Delta\mathbf{n}_f^* = -\nabla_{L_n}(\mathbf{n}_f^*) \tag{21}$$

After removing the normal component from $\Delta\mathbf{n}_f^*$, the new normal vector results from adding the update and then normalizing to satisfy the unit length constraint:

$$\mathbf{n}_f^* \leftarrow \text{normalize}(\mathbf{n}_f^* + (\mathbb{I} - \mathbf{n}_f^* \mathbf{n}_f^{*\top}) \Delta\mathbf{n}_f^*) \tag{22}$$

The Newton step does not distinguish among different types of critical points. We perform a single projected gradient descent step with a fixed step size of 0.1 and normalize in case the direction of the Newton update points away from the gradient after projecting out the normal component $(\nabla_{L_n}(\mathbf{n}_f^*)^\top (\mathbb{I} - \mathbf{n}_f^* \mathbf{n}_f^{*\top}) \Delta\mathbf{n}_f^* < 0)$.

Edges For updating the edges, the ADMM approach leads to the following optimization problem:

$$\arg\min_{\mathbf{e}_{ij}^*} \frac{w_{ij}}{2} \|\mathbf{e}_{ij} - \mathbf{e}_{ij}^*\|^2 + \mu \sum_{f\ni(i,j)} \frac{w_{ij}}{2} \left(u_{fij} + \mathbf{e}_{ij}^{*\top} \mathbf{n}_f^*\right)^2 \tag{23}$$

Here our choice of ρ results in λ being a constant factor that does not affect the optimization. The cotan weights result in the edges having no influence on the computation of the updated edge vectors. This is convenient, because the potentially negative edge weights could otherwise cause edges to rotate against the desired directions. The minimum to the above least squares problem satisfies:

$$\left(\mu \sum_{f\ni(i,j)} \mathbf{n}_f^* (\mathbf{n}_f^*)^\top + \mathbf{I}\right) \mathbf{e}_{ij}^* = \mathbf{e}_{ij} - \mu \sum_{f\ni(i,j)} u_{fij} \mathbf{n}_f^*. \tag{24}$$

The new auxiliary edges $\mathbf{e}_{ij}^*$ can therefore be found as the solution of a linear system.

Dual Variables The update rule for the scale dual variables is:

$$u_{fij} \leftarrow u_{fij} + \mathbf{e}_{ij}^{*\top} \mathbf{n}_f^* \tag{25}$$

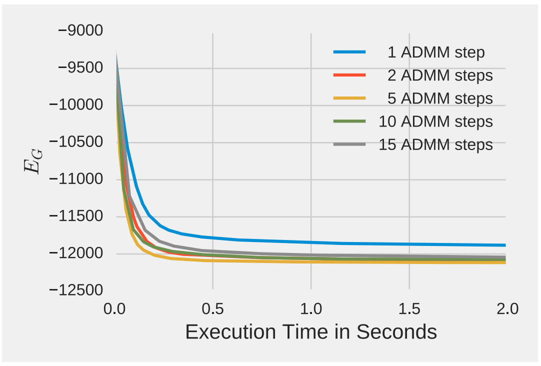

Figure 7: *Energy E_G as a function of the execution time for different numbers of ADMM updates per iteration on a Stanford Bunny down-sampled to* 6172 *vertices.*

Number of ADMM updates While ADMM based methods can take a long time to converge, the approach often leads good results already for a small number of steps [BPC*11, Section 3.2.2]. We observe that the method works well for a range of iteration counts, including a single ADMM update. We have compared the time to numerical convergence of different choices – Figure 7 provides one instance of such an experiment. Notice that energies in the converged state differ, meaning that the number of ADMM steps have influence on the local minimum found by the procedure. Our observation is that fewer ADMM iterations tend to yield results that are closer to the initial geometry, while more iterations give more emphasis on the preferred normals. In fact, the parameter λ offers similar control.

We believe there is no obvious choice for the number of ADMM iterations in practice and, consequently, we expose the iteration count in our implementation to the user (the code is made available with the paper). We prefer to use a single ADMM update. The reason for this choice is that the mesh is updated and ready for the next user interaction in the GUI after each global step. So, quite simply, a single ADMM step provides the highest update rate for user interaction. As there is no technical reason for using more ADMM steps, we favor fluid interaction over a possibly smaller energy or faster convergence to a steady state. All examples in the following sections have been generated with this setting.

3.3. Minimization: Global step

While our choice of energy function introduces no change in the system matrix, the right hand side has to be modified. The partial derivative of equation 6 with respect to $\mathbf{v}_i$ is given by:

$$\frac{\partial E_G^*}{\partial \mathbf{v}_i} = \sum_{(i,j)\in\mathcal{E}} 2w_{ij}(\mathbf{e}_{ij} - \frac{1}{2}(\mathbf{R}_i + \mathbf{R}_j)\hat{\mathbf{e}}_{ij} + \lambda(\mathbf{e}_{ij} - \mathbf{e}_{ij}^*)) \tag{26}$$

Note that $\mathbf{e}_{ij}$ as well as $\mathbf{e}_{ji}$ depend on $\mathbf{v}_i$. Hence, we get a constant factor of $2w_{ij} = 2w_{ji}$. To find the minimum we set this term to zero.

This leads to a formulation very similar to the ARAP energy:

$$\forall i: \sum_{(i,j)\in\mathcal{E}} w_{ij}(\mathbf{e}_{ij} + \lambda \mathbf{e}_{ij}) = \sum_{(i,j)\in\mathcal{E}} w_{ij}\Big(\frac{1}{2}(\mathbf{R}_i + \mathbf{R}_j)\hat{\mathbf{e}}_{ij} + \lambda \mathbf{e}^*_{ij}\Big)$$

$$\Leftrightarrow \sum_{(i,j)\in\mathcal{E}} w_{ij}\mathbf{e}_{ij} = \frac{\Big(\sum_{(i,j)\in\mathcal{E}} \frac{w_{ij}}{2}(\mathbf{R}_i + \mathbf{R}_j)\hat{\mathbf{e}}_{ij}\Big) + \lambda \sum_{(i,j)\in\mathcal{E}} w_{ij}\mathbf{e}^*_{ij}}{1+\lambda} \tag{27}$$

For fixed $\mathbf{R}_i$ and $\mathbf{e}^*_{ij}$, the factor λ can be moved to the right hand side, so that the same constant system matrix as in ARAP is obtained. This allows precomputing the factorization needed to solve the LSE. Constraints on vertex positions can be integrated similarly to ARAP.

4. Implementation

We have extended the implementation of Cubic Stylization made available by the authors [LJ19] and evaluated the performance on an Intel© Core™ i7-4770K processor @ 3.50GHz × 4.

4.1. Algorithm

Algorithm 1 provides pseudo-code for the Gauss stylization update. This can be seen as a replacement for the local and a global update in the classic ARAP algorithm [SA07]. The update is applied repeatedly until the desired effect is achieved. The loops can be easily parallelized with little storage overhead. The right hand side of the global step (Eq. 27) can be computed within the last loop over the edges.

Algorithm 1: Gauss Stylization Update

```
Input  : original vertex positions $\hat{V}$
         current vertex positions $V_n$
         ADMM iterations $N$
Output: updated vertex positions $V_{n+1}$
for $f \in F$ do                                   // Initialization
    $\mathbf{n}^*_f \leftarrow \mathbf{n}_f$
    for $\mathbf{e}_{ij} \in E_f$ do
        $\mathbf{e}^*_{ij} \leftarrow \mathbf{e}_{ij}$, $u_{fij} \leftarrow 0$
for $i \in 1..N$ do                                // ADMM optimization
    for $f \in F$ do
        $\Delta\mathbf{n}^*_f \leftarrow$ newton$(g, \mathbf{n}^*_f, u_{fij})$ (Eq. 21)
        $\mathbf{n}^*_f \leftarrow$ normalize$(\mathbf{n}^*_f + (\Delta\mathbf{n}^*_f - {\Delta\mathbf{n}^*_f}^\mathsf{T}\mathbf{n}^*_f \cdot \mathbf{n}^*_f))$
    for $\mathbf{e}_{ij} \in E$ do
        $\mathbf{e}^*_{ij} \leftarrow$ lssolve$(\mathbf{n}^*_f, \mathbf{e}^*_{ij}, u_{fij})$ (Eq, 24)
    for $f \in F$ do
        for $\mathbf{e}_{ij} \in E_f$ do
            $u_{fij} \leftarrow u_{fij} + {\mathbf{e}^*_{ij}}^\mathsf{T}\mathbf{n}^*_f$
for $v \in \hat{V}$ do                             // local ARAP step
    $R_v \leftarrow$ local$(\hat{V}, V_n)$
$V_{n+1} \leftarrow$ global$(V_n, \{R_i\}, \{\mathbf{e}^*_{ij}\})$ (Eq. 27)        // global step
```

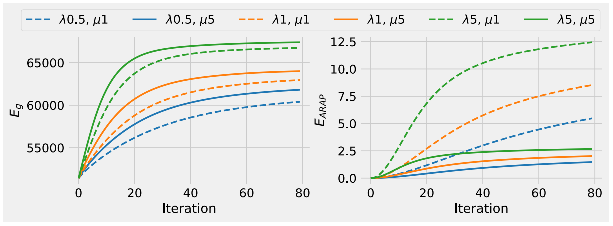

Figure 8: *Graph of the two terms in the energy as set up in Eq. 4. E_{ARAP} starts in a perfect configuration and can only increase. E_g is maximized consistently.*

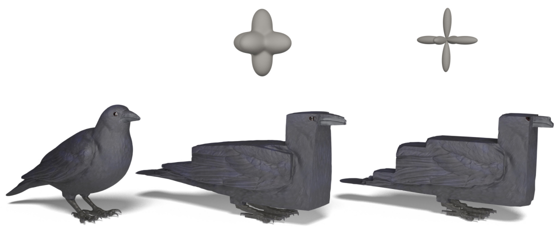

Figure 9: *Different choices of σ lead to different stylistic results. Midlle image uses $\lambda = 21$ and $\sigma = 4.6$, right image $\lambda = 21$ and $\sigma = 21$.© Crow by zixisun02 under CC BY.*

4.2. Optimization and Parameters

Figure 8 shows both components of the energy in Eq. 4 for different parameters on the Stanford Bunny mesh ($E_g = \sum_{f\in\mathcal{F}} g(\mathbf{n}_f)$). Note that the energy contains E_g with a negative sign, the observed increase reflects the desired normal alignment. The λ parameter controls the weighting of the term involving g and thereby the extent of the stylization, clearly visible in both energies. Normal alignment as measured by the g function increases consistently while the ARAP error increases due to the modification of the initial geometry.

Note that despite the different orders of magnitude in the energies, the E_{ARAP} term has a strong influence on the optimization because it is affected proportionally to the magnitude of the gradients. By our design of the g function, its gradient mostly points in direction of the normal, leading to a large part of its magnitude getting lost when projecting out the normal component.

Bigger values for σ lead to faster convergence towards nearby local minima. Figure 9 shows an example. The oblique back of the crow has more jags if σ is larger.

4.3. Performance

In our implementation, a single iteration of the optimization is similar in computational cost to ARAP or Cubic Stylization. Average values for three common meshes are given in Table 1. The values are insensitive to changes in λ, μ, or the geometry of the mesh. Both methods are fast enough to be used whenever ARAP is suitable. Importantly, for many common meshes our implementation is fast enough to provide interactive feedback.

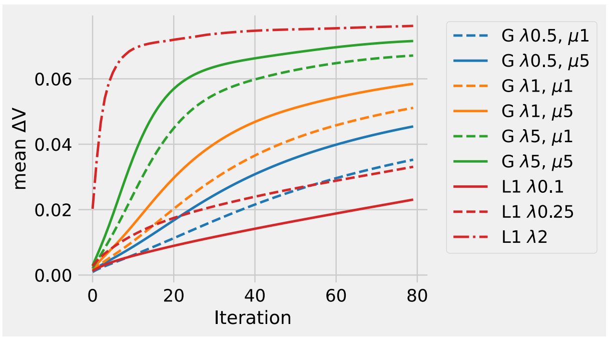

Figure 10: *Mean vertex displacement ΔV to original mesh of the* BUNNY *over the iterations of both Gauss (G) and Cubic (L1) stylization.*

Method	BUNNY ($\sim 35K$)	DRAGON ($\sim 76K$)	ARMADILLO ($\sim 173K$)
Gauss ($\lambda = 4, \sigma = 1$)	76ms	165ms	409ms
Cubic ($\lambda = 2$)	65ms	146ms	342ms

Table 1: *Time per iteration for both Gauss Stylization and Cubic Stylization (as implemented by Liu and Jacobson [LJ19]) on common meshes of different size (approximate vertex count in parentheses). All values are averaged over the first* 50 *iterations.*

The parameters play an important role for the deformation behavior over the iterations. It is not clear that during interactive modeling the converged state, i.e. negligible vertex displacement for further optimization, is relevant. We show a plot of mean vertex displacement in the first iterations in Figure 10. Both Cubic and Gauss stylization can be configured so that approximately 100 iterations yield pleasing results. The local minimum attained during optimization generally depends on the choice of parameters. In our experience, parameter settings that result in slower convergence lead to more global deformation.

5. Results and Evaluation

The energy minimization we propose can be used to apply a variety of different styles to an input mesh and is also well suited for the use together with textures as shown in Figure 11. The resulting models are easy to fabricate using a 3D printer but the reduced number of different surface normals makes it also possible to carve the figure in wood.

We compare with the cubic stylization approach [LJ19] and discuss the additional possibilities and limitations of Gauss stylization.

5.1. Comparison with Cubic Stylization

The symmetric polyhedra used as target shapes in Cubic Stylization [LJ19] can all be formulated as a discrete set of normal constraints, for which Gauss stylization works as well by including two normals per coordinate axis. In Figure 12 we compare the approaches for mesh cubification. The λ parameters have been chosen to achieve a similar effect for both methods. Since our method is face-based, it tends to lead to more prominent edges, the averaged vertex normals in Cubic Stylization [LJ19] tend to smooth the result. However, in a wide range the differences can be compensated for by the choice of the λ parameter.

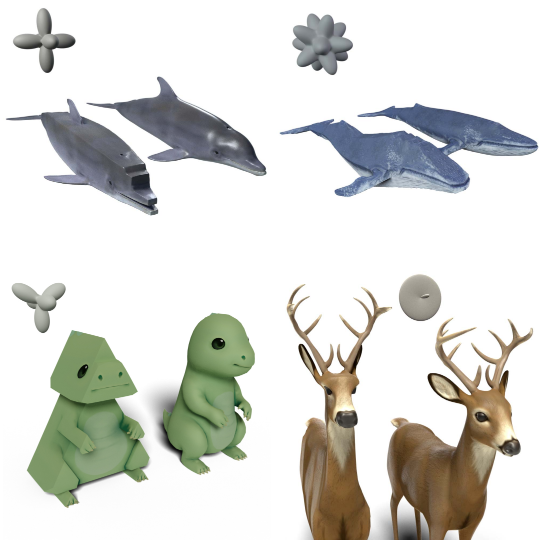

Figure 11: *Gauss stylization enables a variety of different styles. The locally isometric deformations preserve the appearance of the textures. The dolphin is stylized using cube face normals, the whale with dodecahedron normals, the dino with normals of a triangular prism and the deer with normals of a cylinder. © Bottlenose Dolphin by DigitalLife3D, © Blue Whale by Bohdan Lvov, © Dinosaur by zixisun02 and © Little Buck 2.0 by TheCaitasaurus. All under CC BY.*

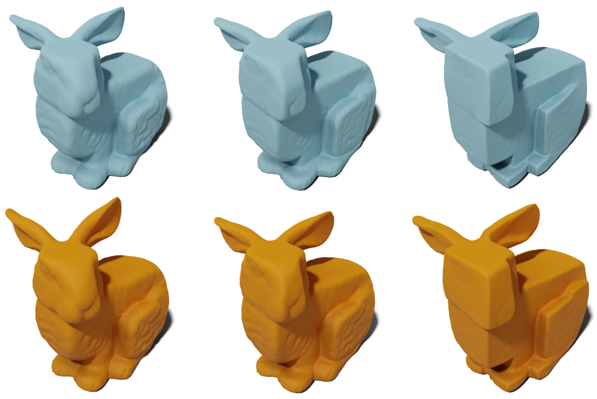

Figure 12: *Stylistic comparison between Gauss and Cubic Stylization. While both methods have their own particular aesthetics, the results are overall visually similar. (top row Cubic Stylization with* $\lambda = 0.1, 0.25, 2$*; bottom row Gauss Stylization with* $\lambda = 0.15, 1, 5$ *(*$\sigma = 8, 3.6$ *and* 4.2*).*

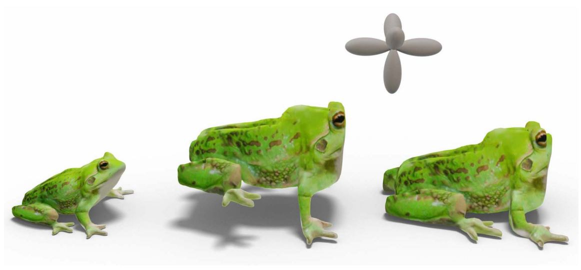

Figure 13: *Different results can be achieved by enforcing positional constraints. For the stylized frog on the right, the constraints are placed such that all feet touch the ground. © Frog by Bogdan Lapitsky under CC BY.*

Artistic controls Cubic Stylization offers a variety of artistic controls that are based on either the properties of ARAP or different parameters of the additional penalty term per vertex. We added a term per face and can provide similar possibilities, except that locally varying parameters need to be defined per face. Figure 13 demonstrates the integration of user-defined positional constraints, i.e. fixed vertices. Locally varying preferred normals, similar to the 'multistyle' n Cubic Stylization, can be achieved by assigning different g functions to different faces – see Figure 14.

5.2. Possibilities of the Gauss Approach

Interactive tool We implemented a graphical user interface that allows us to intuitively model g functions and apply them to different parts of the mesh. Note that modifying g can be done without changing the system matrix, meaning the effects can be explored interactively. As the suggested g function is a linear combination of exponential functions defined by normal directions it is very easy, even for inexperienced users, to model and manipulate the desired normal directions. Figure 15 shows the GUI.

We limit the local modification to μ and g. While it would be possible to also modify λ per edge, this would affect the system matrix and require re-factorization, hindering fluid interaction.

As is common for non convex iterative optimizations our results heavily depend on the initial vertex configuration. The possibility to interactively change g allows modifying it throughout the optimization until a satisfying result is achieved. We feel the visualization of the g function helps abstracting the mathematical details of the optimization and caters to users with little exposure to the underlying theory.

Extreme Cubificiation Cubic Stylization operates on averaged normals and the ℓ_1 term only affects the calculation of the local rotation matrices in the ARAP method. This means the sparsity constraint does not operate directly on edges. Since our method directly constrains the face normals in each triangle, we are able to achieve more extreme cubification effects. Figure 16 (left block) shows several examples of our algorithm for comparatively very high λ values. Similar to the effect of a three dimensional pictogram, the resulting meshes only retain a bare minimum of the original geometry. While they lose a lot of local details, the objects are still clearly distinguishable, making for an interesting visual style.

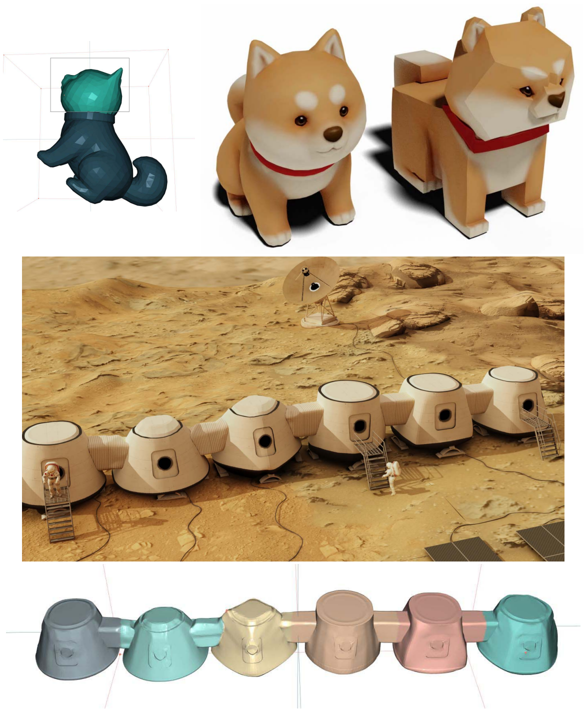

Figure 14: *The preference function g may vary per face – here regions are selected interactively and assigned different preferred normals. The dog has been stylized with doddecahedron normals for the head and cube normals for the body (top row). The station in the Mars scene (middle) is stylized using five different styles (left to right: original, cone, octahedron, cone, truncated pyramid, icosahedron). © Shiba Dog by zixisun02 and © Mars Mission Base by admone under CC BY.*

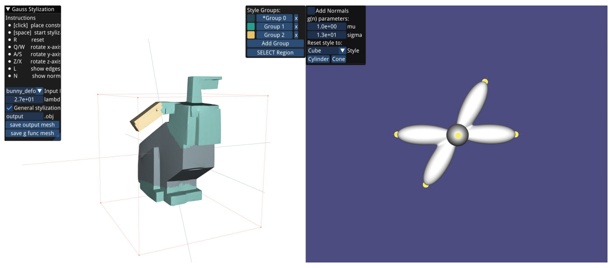

Figure 15: *The graphical user interface shows the object (left) as well as the g function (right). For interactive modeling, the g function can be modified and the optimization is adapted in real time.*

Non-Symmetric Discrete Sets of Normals The restriction to symmetric normals may appear minor, but it does have significant impact on the appearance. Asymmetric shapes, such as the tetrahedron, give a sense of direction, an important visual effect. If a mesh requires additional symmetry constraints (e.g. along an axis

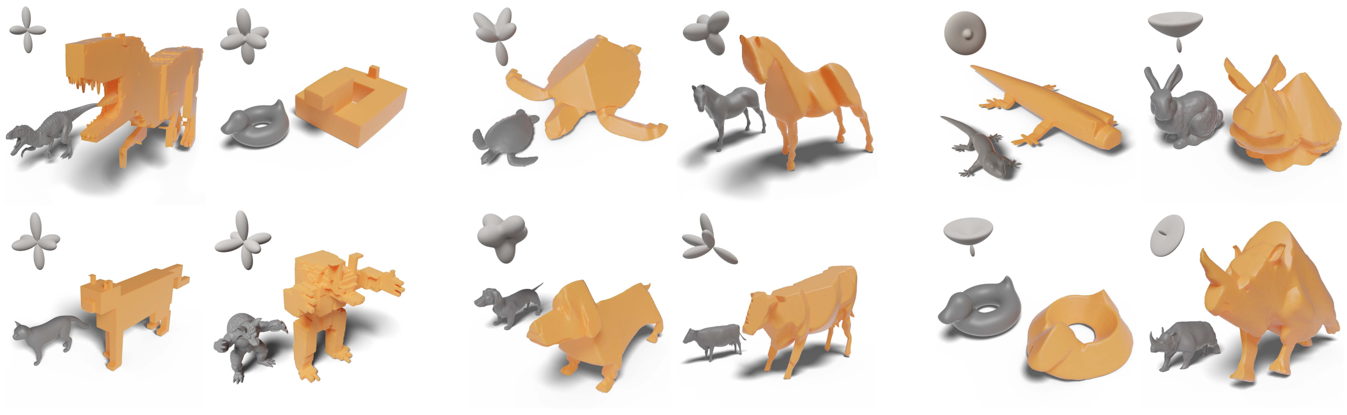

Figure 16: *A variety of stylization results enabled by Gauss Stylization. Left block: higher values of λ lead to 'extreme' cubification - faces almost paralle to those of a cube. Middle block: Preferred sets of normals are not required to be point-symmetric. Right block: Semi-discrete preferred normals allow modeling locally cylinder- or cone-like shapes. © Indominus Rex dinosaur by BlueMesh and © Cat Fripouille by Guillaume Bolis, © Hawksbill Turtle by Bindestrek, © Dog by Pusztai Andras and © Cow by Josué Boisvert, © Tokay Gecko by DigitalLife3D and © Rhino by Gremory Saiyan under CC BY.*

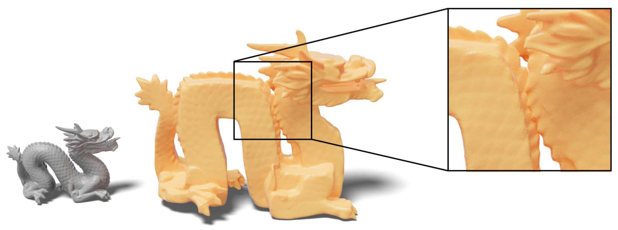

Figure 17: *The deformation of the shape is surface-based, so self-intersection may occur.*

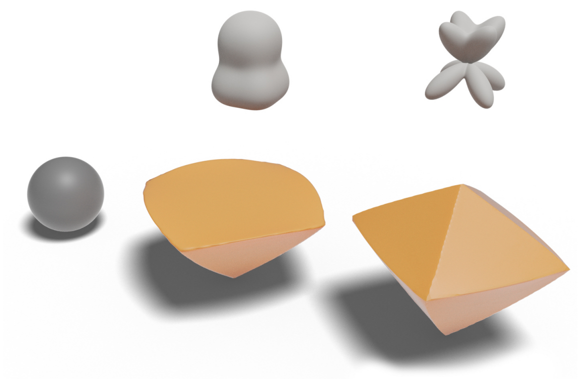

Figure 18: *Example for a malformed g function (left) and the same g function with bigger σ (right).*

such as the body of an animal) or if one side is supposed to stay flat (e.g. to lie on a surface), this further drastically limits the amount of symmetric shapes that can be used. We show examples of the additional freedom in in Figure 16 (middle block).

Semi-Discrete Sets of Normals The approach can further be extended to semi-discrete regions of preferred normal directions. We show results with normals taken from a cylinders or cone in Figure 16. For both, the desired normal directions can be characterized as circles on the Gauss sphere. Since any normal from the circle(s) is a desired direction, the shapes may be curved arbitrarily, i.e. are close to developable but may not be reminiscent of cylinders or cones. Note that it is useful to include the normal direction of the caps of the cylinder or cone to avoid the degeneration of the shape.

Limitations The stylization results depend on the interplay of the mesh geometry, the parameters, and their interactive modification. This may have unwanted effects or unexpected results, as explained below. In particular, the resulting shape is not defined by the choice of parameters at the point of convergence, but rather the whole history of interaction. This might make the reproduction of results difficult.

The strength of deformation depends on the resolution of the mesh. Hence, meshes which consist of differently detailed parts will be stylized inconsistently. Artistic scenes are often build from several meshes that are not necessarily equally detailed. In this case λ should be adjusted locally.

The fact that the maxima of g are not exactly at the defined normal positions can lead to unwanted behavior. A low σ and normals too close to each other may lead to multiple desired maxima merging into a single maximum, as shown in Figure 18). This problem can be remedied by increasing σ.

As the energy is defined on the surface, there is nothing that prohibits the surface from intersecting itself (see Figure 17). Depending on the application, this may or may not be acceptable. Likewise, the volume of the shape is not preserved (cf. Figure 18). A simple way to avoid self intersections or shrinkage is the placing of point constraints in the general ARAP surface modeling framework.

6. Discussion and Conclusions

As documented in the figures , Gauss stylization provides an intuitive and flexible tool for interactive modification of geometric models. It is a natural continuation of the convincing and successful Cubic Stylization [LJ19].

Our technical approach shows that it is possible to provide global control over the surface normals at interactive rates. This opens up more general modeling ideas. Currently, the normal preference function g is defined globally, or for user-selected regions. It is possible to make g dependent on the local surface properties, i.e. normals or curvatures. This would enable a notion of style based on higher order surface statistics, similar to style transfer on images [GEB16]. It may also be used to allow user interaction based on copying such higher order statistics.

Also, we have so far only used analytic preference functions. It might be interesting to let the user freely create g by 'drawing' on the sphere. Likewise, the function g could be generated procedurally. In both cases one could discretize the function on a spherical mesh. Recall that all we need for optimization are first and second derivatives, which may be computed quickly from the discrete representation. The important point for all such extensions is that the computational speed only depends on how quickly g (and its derivatives) can be computed, not that it is constant across the mesh or analytic.

References

[Ale21] ALEXA M.: Polycover: Shape approximating with discrete surface orientation. *IEEE Computer Graphics and Applications 41*, 3 (2021), 85–95. 2

[BCMP18] BICKEL B., CIGNONI P., MALOMO L., PIETRONI N.: State of the art on stylized fabrication. *Computer Graphics Forum 37*, 6 (2018), 325–342. 1

[BDS*12] BOUAZIZ S., DEUSS M., SCHWARTZBURG Y., WEISE T., PAULY M.: Shape-up: Shaping discrete geometry with projections. *Computer Graphics Forum 31*, 5 (2012), 1657–1667. 3

[BH11] BIAN Z., HU S.-M.: Preserving detailed features in digital bas-relief making. *Computer Aided Geometric Design 28*, 4 (2011), 245 – 256. 1

[Bli82] BLINN J. F.: A generalization of algebraic surface drawing. *SIGGRAPH Comput. Graph. 16*, 3 (July 1982), 273. 4

[BPC*11] BOYD S., PARIKH N., CHU E., PELEATO B., ECKSTEIN J.: Distributed optimization and statistical learning via the alternating direction method of multipliers. *Found. Trends Mach. Learn. 3*, 1 (Jan. 2011), 1–122. 5, 6

[CPSS10] CHAO I., PINKALL U., SANAN P., SCHRÖDER P.: A simple geometric model for elastic deformations. In *ACM SIGGRAPH 2010 Papers* (New York, NY, USA, 2010), SIGGRAPH '10, Association for Computing Machinery. 3

[GEB16] GATYS L. A., ECKER A. S., BETHGE M.: Image style transfer using convolutional neural networks. In *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016* (2016), IEEE Computer Society, pp. 2414–2423. 11

[GG01] GOOCH B., GOOCH A.: *Non-photorealistic rendering*. CRC Press, 2001. 1

[HFAT07] HUANG X., FU H., AU O. K.-C., TAI C.-L.: Optimal boundaries for poisson mesh merging. In *Proceedings of the 2007 ACM Symposium on Solid and Physical Modeling* (New York, NY, USA, 2007), SPM '07, Association for Computing Machinery, p. 35–40. 1

[HJS*14] HUANG J., JIANG T., SHI Z., TONG Y., BAO H., DESBRUN M.: ℓ_1-based construction of polycube maps from complex shapes. *ACM Trans. Graph. 33*, 3 (June 2014). 2

[HS13] HE L., SCHAEFER S.: Mesh denoising via l0 minimization. *ACM Trans. Graph. 32*, 4 (July 2013). 3

[Koe90] KOENDERINK J. J.: *Solid shape*. 1990. 1

[LJ19] LIU H.-T. D., JACOBSON A.: Cubic stylization. *ACM Trans. Graph. 38*, 6 (Nov. 2019). 1, 2, 3, 4, 7, 8, 11

[LYH*15] LUO S.-J., YUE Y., HUANG C.-K., CHUNG Y.-H., IMAI S., NISHITA T., CHEN B.-Y.: Legolization: Optimizing lego designs. *ACM Trans. Graph. 34*, 6 (Oct. 2015). 2

[MDSB03] MEYER M., DESBRUN M., SCHRÖDER P., BARR A. H.: Discrete differential-geometry operators for triangulated 2-manifolds. In *Visualization and Mathematics III* (Berlin, Heidelberg, 2003), Hege H.-C., Polthier K., (Eds.), Springer Berlin Heidelberg, pp. 35–57. 3

[PP93] PINKALL U., POLTHIER K.: Computing discrete minimal surfaces and their conjugates. *Experim. Math. 2* (1993), 15–36. 3

[SA07] SORKINE O., ALEXA M.: As-rigid-as-possible surface modeling. In *Proceedings of the Fifth Eurographics Symposium on Geometry Processing* (Goslar, DEU, 2007), SGP '07, Eurographics Association, p. 109–116. 1, 2, 3, 7

[SBS07] SONG W., BELYAEV A., SEIDEL H.: Automatic generation of bas-reliefs from 3d shapes. In *IEEE International Conference on Shape Modeling and Applications 2007 (SMI '07)* (2007), pp. 211–214. 1

[SGC18] STEIN O., GRINSPUN E., CRANE K.: Developability of triangle meshes. *ACM Trans. Graph. 37*, 4 (July 2018). 2

[SGW06] SCHMIDT R., GRIMM C., WYVILL B.: Interactive decal compositing with discrete exponential maps. In *ACM SIGGRAPH 2006 Papers* (New York, NY, USA, 2006), SIGGRAPH '06, Association for Computing Machinery, p. 605–613. 1

[SPSH14] SCHÜLLER C., PANOZZO D., SORKINE-HORNUNG O.: Appearance-mimicking surfaces. *ACM Trans. Graph. 33*, 6 (Nov. 2014). 1

[SS10] SCHMIDT R., SINGH K.: Meshmixer: An interface for rapid mesh composition. In *ACM SIGGRAPH 2010 Talks* (New York, NY, USA, 2010), SIGGRAPH '10, Association for Computing Machinery. 1

[Str02] STROTHOTTE T.: *Non-photorealistic computer graphics : modeling, rendering, and animation / Thomas Strothotte, Stefan Schlechtweg.* Morgan Kaufmann, San Francisco, Calif., 2002. 1

[THCM04] TARINI M., HORMANN K., CIGNONI P., MONTANI C.: Polycube-maps. In *ACM SIGGRAPH 2004 Papers* (New York, NY, USA, 2004), SIGGRAPH '04, Association for Computing Machinery, p. 853–860. 2

[TSP13] TESTUZ R. P., SCHWARTZBURG Y., PAULY M.: Automatic generation of constructable brick sculptures. *Eurographics 2013-Short Papers* (2013), 81–84. 2

[TSS*11] TAKAYAMA K., SCHMIDT R., SINGH K., IGARASHI T., BOUBEKEUR T., SORKINE O.: Geobrush: Interactive mesh geometry cloning. *Computer Graphics Forum 30*, 2 (2011), 613–622. 1

[Wat82] WATSON G. S.: Distributions on the circle and sphere. *Journal of Applied Probability 19*, A (1982), 265–280. 4

[WDB*07] WEYRICH T., DENG J., BARNES C., RUSINKIEWICZ S., FINKELSTEIN A.: Digital bas-relief from 3d scenes. *ACM Trans. Graph. 26*, 3 (July 2007), 32–es. 1

[WFL*15] WANG P.-S., FU X.-M., LIU Y., TONG X., LIU S.-L., GUO B.: Rolling guidance normal filter for geometric processing. *ACM Trans. Graph. 34*, 6 (Oct. 2015). 3

[ZDH*19] ZHANG J., DENG B., HONG Y., PENG Y., QIN W., LIU L.: Static/dynamic filtering for mesh geometry. *IEEE Transactions on Visualization and Computer Graphics 25*, 4 (2019), 1774–1787. 3