IGS2019 International Geometry Summit

Posters' Proceedings

19th June 2019

Vancouver, Canada

In June 2019, the **IGS2019 - International Geometry Summit** features 4 major conferences in Computer Graphics and applications

- **SPM** - Solid and Physical Modelling
- **SIAM/GD** Computational Geometric Design
- **SMI** - Shape Modelling International
- **GMP**- Geometric Modelling and Processing

IGS2019 posters describe recent work, highly relevant results of work in progress, successful systems, and applications, in all areas related to

- solid and physical modelling
- computational geometric design
- shape modelling and analysis
- geometric modelling and processing

After a review phase, 20 poster presentations were accepted to the program, and they were presented during a joint poster session of IGS2019 that offered interactive discussion between presenters and attendees. A fast-forward oral session was held at the beginning of the program where authors presented a brief summary of their work to all IGS2019 attendees.

Special thanks are given to the IGS2019 Chairs, Konrad Polthier, Wenping Wang, and Richard (Hao) Zhang for their support to the IGS2019 Poster Session.

IGS2019 Poster Chairs

- Carolina Beccari (University of Bologna, Italy)
- Xin Shane Li (Louisiana State University, USA)
- Tao Ju (Washington University, USA)
- Giuseppe Patanè (CNR-IMATI, Italy)

Posters

- **Mesh denoising via half-kernel Laplacian**, Wei Pan, Xuequan Lu, Yuanhao Gong, Ying He, Guoping Qiu
- **Scalable coding of dynamic 3D meshes for low-latency streaming applications**, Arvanitis Gerasimos, Aris S. Lalos, Konstantinos Moustakas
- **Control of lines of curvature for plate forming in shipbuilding**, Masahito Takezawa, Kohei Matsuo, Takashi Maekawa
- **Detection of discontinuities from scattered data**, Cesare Bracco, Oleg Davydov, Carlotta Giannelli, Alessandra Sestini
- **EdgeNet: deep metric learning for 3D shapes**, Mingjia Chen, Qianfang Zou, Changbo Wang, Ligang Liu
- **Mesh learning using persistent homology on the Laplacian eigenfunctions**, Yunhao Zhang, Haowen Liu, Paul Rosen, and Mustafa Hajij
- **3D Femur skeletonization using maximum-minimum centre approach**, Saw Seow Hui, Ewe Hong Tat, Ji Wan Kim, Byung Gook Lee
- **Triangular trigonometric patches for surface interpolant**, Xiang Fang, Stephen Mann
- **Quasi arc-length approximation with cubic B-splines**, Javier Sa ́nchez-Reyes, Jesu ́s M. Chacon, Ruben Dorado
- **Two-scale mechanical design with effective material property envelope**, Xingchen Liu
- **Geometrically smooth Catmull-Clark spline surfaces**, Ahmed Blidia, Bernard Mourrain
- **3D Reconstruction using 2D triangulation,** Florian Gawrilowicz, J. Andreas Bærentzen
- **CT-shape: coordinated triangle based reconstruction from dot patterns and boundary samples**, Safeer Babu Thayyil, Amal Dev Parakkat, Ramanathan Muthuganapathy
- **Scikit-Shape: python toolbox for shape analysis and segmentation**, Gunay Dogan
- **Variational shape approximation of point set surfaces**, Martin Skrodzki, Eric Zimmermann, Konrad Polthier
- **An optimized Yarn-Level geometric model for FEA simulation of Weft-Knitted fabrics**, Paras Wadekar, Eric Markowicz, Dani Liu, Genevieve Dion, Antonios Kontsos, David Breen
- **Positive geometries for barycentric interpolation**, Marton Vaitkus
- **CHoCC: convex hull of cospherical circles**, Yaohong Wu, Ashish Gupta, Kelsey Kurzeja, Jarek Rossignac
- **Computing intersection areas of overlaid 2D meshes**, W. Randolph Franklin, Salles Viana Gomes de Magalhães
- **Spline-based interactive object segmentation using SplineRNN**, Ivar Stangeby, Oliver Barrowclough, Georg Muntingh

Mesh Denoising via Half-kernel Laplacian

Wei Pan [1], Xuequan Lu [2], Yuanhao Gong [1], Ying He [3], and Guoping Qiu [1]

[1] *College of Information Engineering, Shenzhen University, Guangdong Key Laboratory of Intelligent Information Processing*
[2] *School of Information Technology, Deakin University*
[3] *Nanyang Technological University*

Abstract

We propose a half-kernel Laplacian operator (HLO) for feature-preserving mesh denoising. We develop a half-window algorithm to divide the neighborhood of each vertex into paired subsets (half windows), and compute the discrete Laplacians with uniform weights of all subsets. We determine the final half-kernel Laplacian as the one incurring the least regularization energy. To remove noise, we update vertices with the determined Laplacians in an iterative manner. Our approach, involving a single parameter (i.e., the number of iterations for vertex update), is conceptually simple and easy to implement. We show that the proposed HLO can better preserve features and avoid the shrinkage artifact than the uniform Laplacians. Experiments demonstrate that our results are better or comparable to the state-of-the-art techniques, in terms of visual examination and quantitative measure.

1 Introduction

Surface meshes acquired through scanning or sensing equipment are inevitably contaminated with noise. These meshes have to be processed with denoising techniques [4, 5, 3, 8, 9] before applying to other fields like computer animation. The main technical challenge of mesh denoising is preserving features while removing noise and alleviating shrinkage. The design of robust mesh denoising methods is therefore particularly needed in nowadays.

The discrete Laplacian operator on triangle meshes has been proven highly useful in digital geometry processing, for example, mesh fairing, surface parameterization, reconstruction and editing [7, 1, 6]. The differential surface representation encodes rich local information such as the mean curvature and the orientation. Despite that the uniform Laplacian operator can effectively smooth surfaces, it fails to preserve features and often leads to shrinkage. In this paper, we propose a novel, robust approach for feature-preserving mesh denoising. Our key idea is to construct a “half-kernel” uniform Laplacian operator (HLO) that can approximate the Laplacians at feature/non-feature points using half windows. Given a noisy mesh as input, our approach can automatically produce a quality feature-preserved mesh without shrinkage.

Our method is conceptually simple and easy to implement, since it involves *a single parameter* (i.e., the number of iterations for vertex update). We demonstrate that the proposed HLO substantially outperforms the uniform Laplacian in preserving features and avoiding shrinkage. Experiments show that our approach achieves comparable or better results to the state-of-the-art mesh denoising techniques.

2 Methodology

The proposed operator is to use half windows to approximate the Laplacians at vertices. To produce half windows, each of the immediate neighboring vertex of the current vertex will be paired with the other unique neighboring vertex to partition the neighborhood into two half windows. The other unique neighbor paired by the starting neighbor is determined as the neighboring vertex which has the shortest distance to the plane defined by the current vertex, the starting neighbor and the centroid of the neighborhood. As illustrated in Fig. 1a, regarding the current vertex v_0, v_D is chosen as the other unique neighbor and paired with the starting neighbor v_A. v_0' is the centroid of the neighborhood. As a result, each vertex v_i and its neighborhood has $|NV(v_i)|$ partition choices which further generate $2|NV(v_i)|$ subsets (half windows).

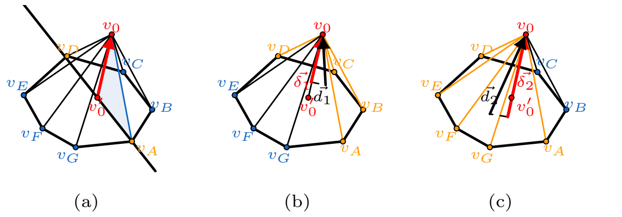


Figure 1: Half-kernel Laplacian on v_0. (a) v_A, the starting neighbor of v_0, is paired with the other unique neighbor v_D which has the shortest distance to the plane $v_0 v_A v_0'$. v_0' is the centroid of the neighbors. The line $v_A v_D$ partitions the neighborhood into the left and right half windows. (b) and (c): d_1 and d_2 are computed by performing the uniform Laplacian to the half windows. δ_1 and δ_2 are achieved by projecting d_1 and d_2 to the full-window uniform Laplacian, respectively.

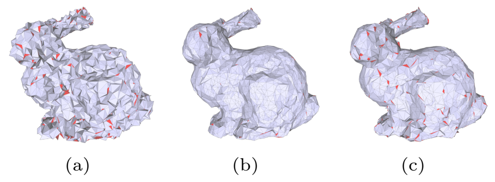


Figure 2: (a) Noisy bunny. (b) 1 iteration of vertex update with projection. (c) 1 iteration of vertex update without projection. Flipped triangles are rendered in red.

We apply the uniform Laplacian operator independently to each of these subsets and compute half-kernel Laplacians for each vertex v_i. Directly using the half-kernel Lapalcians may result in poor results such as degenerating and/or flipping triangles (Fig. 2(c)). We thus project them onto the full-kernel Laplacians to obtain the intermediate half-kernel Laplacians. Fig. 1(b-c) shows two such examples. We can easily enumerate all the intermediate half-kernel Laplacians for v_i: $\dots, \delta_L(v_k), \delta_R(v_k), \dots$. $\delta_L(v_k)$ and $\delta_R(v_k)$ respectively denote the half-kernel Laplacians for the left subset and right subset with the starting neighbor v_k ($\in NV(v_i)$). We next need to determine the optimal Laplacian among the $2|NV(v_i)|$ half-kernel Laplacians. The optimal one will be selected based on the regularization energy [2]. In this work, we define the regularization energy of each vertex as the sum of the norm of the Laplacian and the the distance to the original vertex position, shown as follows:

$$\mathbb{E}(\delta_i^t) = \|\delta_i^t\| + \|v_i^t - v_i^0\|, \tag{2.1}$$

where δ_i^t can be any among the calculated $2|NV(v_i)|$ half-kernel Laplacians in the t-th iteration. v_i^t is the position of vertex v_i in the t-th iteration and v_i^0 is the initial position of v_i. The first term represents the movement between two consecutive iterations, and the second measures the distance between the position at the t-th iteration and the initial position of vertex v_i. We compute the energy $\mathbb{E}$ for each half-kernel Laplacian of vertex v_i and select the optimal Laplacian that incurs the least energy. The vertex position v_i^{t+1} in the $(t+1)$-th iteration is updated by v_i^t and the determined δ_i^t in the t-th iteration.

$$v_i^{t+1} = v_i^t + \delta_i^t \tag{2.2}$$

3 Results

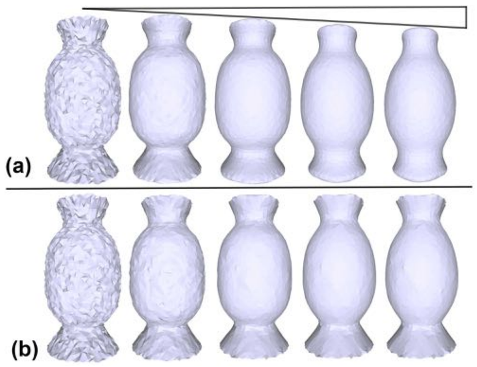


Figure 3: Denoising results for (a) the uniform Laplacian operator and (b) the half-kernel Laplacian operator (HLO). From left to right: The original noisy shape, the result after 1, 3, 5 and 15 iterations; The wedge labels the shrinking effect caused by the uniform Laplacian operator.

Fig. 3 compares our HLO with the uniform Laplacian. It shows that HLO can better preserve features and resist shrinkage than the uniform Laplacian. Fig. 4 shows the denoising results of two synthetic models and a scanned model. The results demonstrate that our method is better or comparable to state-of-the-art mesh denoising techniques.

4 Conclusion

We introduced a novel method for feature-preserving mesh denosing. We developed a Half-kernel Laplacian operator (HLO) which can effectively preserve features and avoid the shrinkage artifact. Experiments show that our approach achieves comparable or better results (quality and quantity) to state-of-the-art mesh denoising techniques.

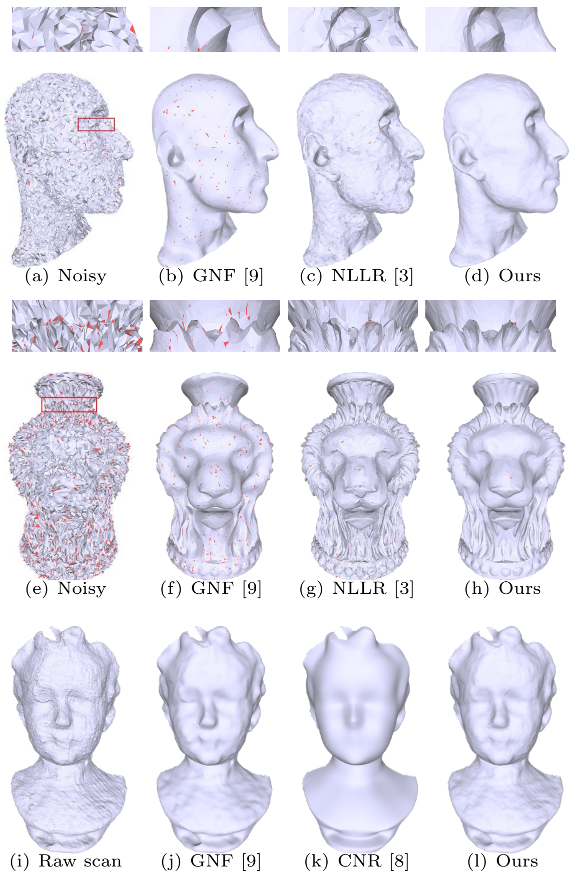


Figure 4: Results on 2 models with synthetic noise (rows 1 and 2) and a scanned model (row 3). The mean square angular errors ($\times 10^{-2}$) [5] are (from 2nd left to right): 58.15, 38.44, 10.33 for row 1; 166.6, 82.25, 32.35 for row 2; 8.50, 9.70, 8.06 for row 3. The corresponding positional errors E_v ($\times 10^{-3}$) [4] are: 0.93, 0.92, 1.08; 13.98, 11.29, 15.87; 5.06, 4.95, 4.86. Flipped triangles are colored in red.

References

[1] Michael S. Floater and Kai Hormann. Surface parameterization: a tutorial and survey. In Neil A. Dodgson, Michael S. Floater, and Malcolm A. Sabin, editors, *Advances in Multiresolution for Geometric Modelling*, pages 157–186, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg.

[2] Yuanhao Gong and Ivo F. Sbalzarini. Curvature filters efficiently reduce certain variational energies. *IEEE Transactions on Image Processing*, 26(4):1786–1798, April 2017.

[3] Xianzhi Li, Lei Zhu, Chi-Wing Fu, and Pheng-Ann Heng. Non-local low-rank normal filtering for mesh denoising. *Computer Graphics Forum*, 37(7):155–166, 2018.

[4] X. Lu, Z. Deng, and W. Chen. A robust scheme for feature-preserving mesh denoising. *IEEE Transactions on Visualization and Computer Graphics*, 22(3):1181–1194, March 2016.

[5] Xuequan Lu, Wenzhi Chen, and Scott Schaefer. Robust mesh denoising via vertex pre-filtering and l1-median normal filtering. *Computer Aided Geometric Design*, 54:49 – 60, 2017.

[6] Olga Sorkine. Differential representations for mesh processing. In *Computer Graphics Forum*, volume 25, pages 789–807. Wiley Online Library, 2006.

[7] G TaubinÝ. Geometric signal processing on polygonal meshes. In *In Proceedings of EUROGRAPHICS 2000: State of the Art Report (STAR)*, 2000.

[8] Peng-Shuai Wang, Yang Liu, and Xin Tong. Mesh denoising via cascaded normal regression. *ACM Transactions on Graphics (SIGGRAPH Asia)*, 35(6), 2016.

[9] Wangyu Zhang, Bailin Deng, Juyong Zhang, Sofien Bouaziz, and Ligang Liu. Guided mesh normal filtering. *Comput. Graph. Forum*, 34(7):23–34, October 2015.

Scalable Coding of Dynamic 3D Meshes for Low-latency Streaming Applications

Arvanitis Gerasimos [1], Aris S. Lalos [2], and Konstantinos Moustakas [1]

[1] *Electrical and Computer Engineering, University of Patras, Greece*
[2] *Industrial Systems Institute, "ATHENA" Research Center*

Abstract

Recently, 3D visual representations of highly deformable 3D models, such as dynamic 3D meshes, are becoming popular due to their capability to represent realistically the motion of real-world objects/humans, paving the road for new and more advanced immersive virtual, augmented and mixed reality experiences. However, the real-time streaming of such models introduces increasing challenges related to low cost, low-latency and scalable coding (SC) of the acquired information. This article proposes an efficient SC mechanism, that decomposes a mesh sequence into spatial and temporal layers that remove a single vertex at each layer. The removed vertices are predicted by performing Laplacian interpolation (LI) of the motion vectors. The artifacts that are introduced in low-resolution representations are mitigated using a subspace based normal-vector denoising procedure, that is optimized to support low-latency streaming scenarios using incremental SVD (ISVD). Additionally, a novel initialization strategy offers robustness to outliers generated due to local deformations.

1 Introduction

3D meshes are widely used in various applications in different scientific fields from heritage science and education to health and robotics. Recently, the interest in 3D mesh sequences has also been increased because of the rapid growth of new 3D scanning technologies, 3D cinema/television, and immersive telepresence systems capable to provide VR/AR experiences. Streaming of 3D animated objects can be used in applications where the geometric data is live-captured and needs to be available in real-time. Nevertheless, these types of applications demand the storage and the transmission of a huge amount of 3D data. The real-time rendering of 3D models, representing either real-world or synthetic objects, generates massive datasets and it requires the use of efficient and fast algorithms for increasing the compression ratios without affecting noticeably the visual quality of the object. For this reason, various techniques on dynamic 3D mesh processing should be developed to address the growing demand and at the same time, to handle these important challenges. Despite the fact that geometry data are generally encoded in a lossy manner, SC seems to be the most promising approach especially in cases where the network performance is unstable. Additionally, a stringent latency is a vital requirement for providing a pleasant immersive VR/AR experience. This means that any real-transmitted dynamic 3D mesh must be easily perceivable, at any frame of its sequence, without affecting its visual quality regardless if a decrease in the transmission rate takes place at any moment. The main purpose of low-latency applications is avoiding to disturb the user's perception because this can negatively affect the quality of the experience. In this work, we take into account all the aforementioned challenges in order to develop a SC method for reliable streaming of dynamic 3D meshes ideal for low-latency streaming applications. Our research counts on the observation that a reduced frame of a mesh sequence can efficiently be reconstructed by taking advantage of the general spatiotemporal information of the entire animated mesh. The proposed method is scalable, since a different number of points are transmitted in each frame, depending on the network capability. To summarize, the main contributions of this work are:

- We propose a mechanism for decomposing a mesh sequence into layers that remove a single vertex at each layer. This decomposition is based on (i) topological characteristics of the mesh and (ii) the temporal behavior of each point separately as their position change through the time frames. In this way, we remove vertices that can be predicted accurately by their neighbors.
- We introduce a process for the online reconstruction of the removed vertices per frames by exploiting coherences on the subspaces corresponding to the different layered representations of the corresponding normals.
- The final reconstructed model has the same number of vertices as the original. Additionally, the method is totally parameter-free and no modification or exhaustive searching of ideal values are required.

2 Overview of our Method

Fig. 1 briefly presents the proposed framework, highlighting the most important procedures of our approach. We start with the layer decomposition process taking into account the spatial and temporal information of the dynamic mesh. The output of this process corresponds to the active points at the end of the removal process. After the transmission, each reduced frame is reconstructed. Firstly, we use a weighted LI approach, as a coarse reconstruction process, in order to estimate the position of the removed vertices. Then, we perform a fine estimation step by tracking the normals subspace deviation between different layers using ISVD, significantly reducing the required complexity as compared to a conventional SVD based approach. The convergence of this approach is significantly accelerated using Robust PCA (RPCA) as an initialization procedure. Finally, each frame is fine-reconstructed penalizing displacement of the vertices over a tangent plane perpendicular to the local surface normal.

2.1 Layer Decomposition

For the SC of a mesh sequence, we propose a spatiotemporal layer decomposition algorithm [1], which removes the vertices of a mesh taking into account both topological and temporal criteria. We assume that only one vertex is removed at each layer so that k spatial layers are created. We annotate as $\mathbf{M}_1$

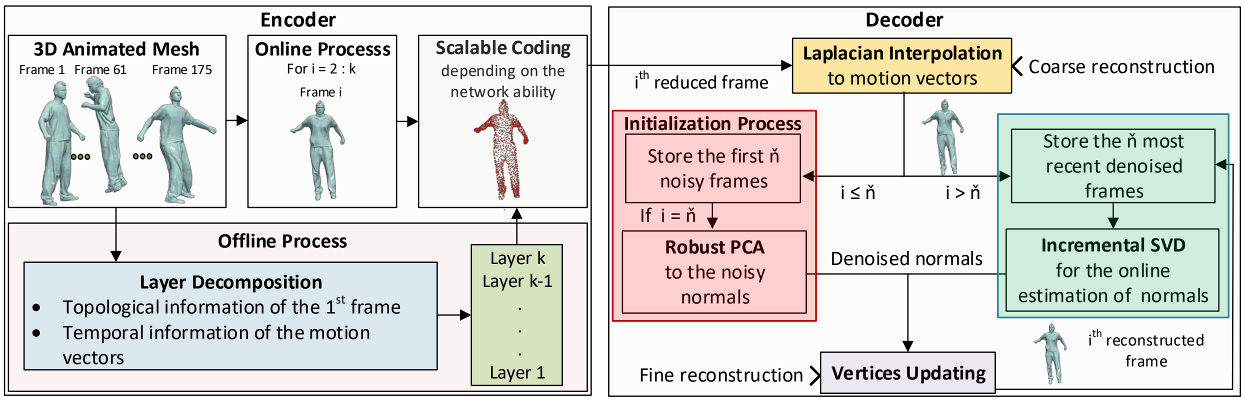

Encoder
3D Animated Mesh
Frame 1 Frame 61 Frame 175
Online Processs
For i = 2 : k
Frame i
Scalable Coding depending on the network ability
Offline Process
Layer Decomposition
Topological information of the 1st frame
Temporal information of the motion vectors
Layer k
Layer k-1
Layer 1
Decoder
ith reduced frame
Laplacian Interpolation to motion vectors
Coarse reconstruction
Initialization Process
Store the first ň noisy frames
If i = ň
Robust PCA to the noisy normals
i ≤ ň
i > ň
Store the ň most recent denoised frames
Incremental SVD for the online estimation of normals
Denoised normals
ith reconstructed frame
Fine reconstruction
Vertices Updating

Figure 1: The pipeline of the proposed method.

the set of points of the layer 1 which has only one vertex while in the highest layer k there is the set of points $\mathbf{M}_k$ consisting of k vertices. The relation between the set of points $\mathbf{M}_l$ and the exactly previous set $\mathbf{M}_{l-1}$ can be described as:

$$\mathbf{M}_l = \mathbf{M}_{l-1} \cup \{v\} \tag{2.1}$$

where $\mathbf{M}_l \subseteq \mathbf{M}\ \forall\ l = 2 \cdots k$. The decomposition process is repeated k times until only one vertex will have remained in layer 1. At each layer, we remove this specific vertex which can be efficiently predicted by the reconstruction process. The proposed cost function consists of two terms, namely the spatial C_s and the temporal C_t:

$$C(i,l) = C_s(i,l) + \lambda C_t(i) \tag{2.2}$$

where $\lambda = 0.1$. The removed vertex v_l at layer l is given by:

$$v_l = \underset{v \in M_l}{\arg\min}\, C(i,l) \tag{2.3}$$

where $C(i,l)$ is the removal cost of vertex i at layer l.

2.1.1 Vertex Removal Using Topology Information

The spatial term C_s is related to the geometry of the first frame and it is estimated as follows. We define as $R_j(i)$ the set of the j-ring neighbors of the i vertex and the $\hat{R}_j(i,l)$ as a subset of $R_j(i)$ with the remaining j-ring neighbors of i vertex in the l level. The spatial term C_s is estimated as:

$$C_s(i,l) = \sum_{j=1}^{3} \frac{|R_j(i)| - |\hat{R}_j(i,l)|}{|R_j(i)|} \rho^j \tag{2.4}$$

where $|.|$ operator returns the number of elements in a set and ρ is a positive parameter.

2.1.2 Vertex Removal Using Temporal Information

Generally, a stationary or slow-motioned point is more likely to be accurately predicted. On the other hand, highly deformable surface patches are less accurately predictable. According to this observation, we propose the use of the temporal term C_t that exploits the temporal information from frame to frame, taking into account the mean motion vector of each point:

$$C_t(i) = \frac{\sum_{t=2}^{n} \|\mathbf{v}_i(t) - \mathbf{v}_i(t-1)\|_2}{n}\ \forall\ i = 1 \cdots k \tag{2.5}$$

where (t) represents the current frame while the (t-1) represents the previous frame.

2.2 Coarse Reconstruction via LI

2.2.1 Weighted Graph Laplacian Matrix

The binary Laplacian matrix provides information regarding the connectivity of vertices. In order to set the preferable constraints, we construct a modified weighted Laplacian matrix, similar to [3], that takes into account the two following factors $\mathbf{H}$ and $\mathbf{B}$. Parameter $\mathbf{H}$ is related to the distance between connected vertices according to:

$$\mathbf{H}_{ij} = \begin{cases} \frac{1}{\|\mathbf{v}_i(t-1) - \mathbf{v}_j(t-1)\|_2} & \text{if } j \in R_1(i) \\ 0 & \text{otherwise} \end{cases} \quad \forall\ i = 1 \cdots k \tag{2.6}$$

For the estimation of this parameter, the values of the vertices from the previously reconstructed frame $(t-1)$ are used. Parameter $\mathbf{B}$ is related to the connecting proximity b (degree or topological distance) of an unknown vertex with an already known vertex. The initial known vertices have a value equal to 4 (reinforcing the contribution of the known values), while the value of the unknown vertices depends on their connectivity degree b, as shown in the following equation:

$$\mathbf{B}_{ij} = \begin{cases} 4\mathbf{A}_{ij} & \text{if } \mathbf{v}_i \text{ is known} \\ \frac{\mathbf{A}_{ij}}{b+1} & \text{otherwise} \end{cases} \quad \forall\ i,j = 1 \cdots k \tag{2.7}$$

where $\mathbf{A}$ is the binary adjacency matrix. Finally, the weighted adjacency matrix $\mathbf{A}_w$ is estimated according to:

$$\mathbf{A}_w = \mathbf{H} \circ \mathbf{B} \circ \mathbf{A} \tag{2.8}$$

where $\circ$ denotes the Hadamard product. Then, the weighted Laplacian matrix is estimated according to:

$$\mathbf{L}_w = \mathbf{D} - \mathbf{A}_w \tag{2.9}$$

where $\mathbf{D} = diag\{D_1, \ldots, D_k\}$ is a diagonal matrix with $D_i = \sum_{j=1}^{k} \mathbf{A}_{w_{ij}}$.

2.2.2 Weighted Laplacian Interpolation

We follow the same line of thought with [9] but applying it on the motion vectors $\boldsymbol{\delta}$ [2] of the vertices instead of the vertices directly.

$$\boldsymbol{\delta}_i = [\delta_{xi},\ \delta_{yi},\ \delta_{zi}]^\intercal \begin{cases} \delta_{xi} = |v_{xi}(t) - v_{xi}(t-1)| \\ \delta_{yi} = |v_{yi}(t) - v_{yi}(t-1)| \\ \delta_{zi} = |v_{zi}(t) - v_{zi}(t-1)| \end{cases} \quad \forall\ i = 1 \cdots k \tag{2.10}$$

Additionally, we use the estimated weighted Laplacian matrix $\mathbf{L}_w$ of Eq. (2.9) which encloses all the necessary constraints for an efficient weighted Laplacian interpolation. We also define $\mathbf{d} = [\boldsymbol{\delta}_1 \ \boldsymbol{\delta}_2 \ \cdots \ \boldsymbol{\delta}_k] \in \mathbb{R}^{k\times 3}$ the matrix which represents the motion vectors of each vertex of the mesh. The Laplacian of $\mathbf{d}$ is written as: $\Delta\boldsymbol{\delta} = \mathbf{L}_w\mathbf{d}$. Next, we split the $\mathbf{d}$ into two parts: $\mathbf{d_k} \in \mathbb{R}^{k'\times 3}$ containing the motion vectors of known vertices and $\mathbf{d_u} \in \mathbb{R}^{k-k'\times 3}$ containing zeros because of the unspecified values of the unknown vertices. Please note that $k-k'$ is equal to the decomposition layer. Correspondingly, the $\mathbf{L}_w$ can be partitioned into four parts: $\mathbf{L}_w = \begin{pmatrix}\mathbf{L}_{w_{11}} & \mathbf{L}_{w_{12}}\\ \mathbf{L}_{w_{21}} & \mathbf{L}_{w_{22}}\end{pmatrix}$, $\mathbf{L}_{w_{11}} \in \mathbb{R}^{k'\times k'}$, $\mathbf{L}_{w_{12}} \in \mathbb{R}^{k'\times k-k'}$, $\mathbf{L}_{w_{21}} \in \mathbb{R}^{k-k'\times k'}$, $\mathbf{L}_{w_{22}} \in \mathbb{R}^{k-k'\times k-k'}$. The Euclidean norm $|\Delta\boldsymbol{\delta}|$ is minimized:

$$\left|\begin{pmatrix}\mathbf{L}_{w_{11}} & \mathbf{L}_{w_{12}}\\ \mathbf{L}_{w_{21}} & \mathbf{L}_{w_{22}}\end{pmatrix}\begin{pmatrix}\mathbf{d_k}\\ \mathbf{d_u}\end{pmatrix}\right| = \left|\begin{pmatrix}\mathbf{L}_{w_{11}}\\ \mathbf{L}_{w_{21}}\end{pmatrix}\mathbf{d_k} + \begin{pmatrix}\mathbf{L}_{w_{12}}\\ \mathbf{L}_{w_{22}}\end{pmatrix}\mathbf{d_u}\right| \tag{2.11}$$

By solving this system we estimate the unknown motion vectors $\mathbf{d_u}$. The coordinates of the missing vertices are estimated by updating their position using the estimated $\mathbf{d_u}$.

$$\mathbf{v_u}(t) = \mathbf{v_u}(t-1) + \mathbf{d}_u, \ \ \forall\ t = 2\cdots n \tag{2.12}$$

Finally, all vertices of the incomplete frame (t) are known $\mathbf{v}(t) = \mathbf{v_k}(t) \cup \mathbf{v_u}(t)$ where $\mathbf{v_k} = [\mathbf{v_{k}}_1 \cdots \mathbf{v_{k}}_{k'}]$ and $\mathbf{v_u} = [\mathbf{v_u}_{k'+1} \cdots \mathbf{v_u}_{k-k'}]$.

2.3 Online SC Using Fine Reconstruction

The previously presented coarse reconstructed step demonstrates impressive performance. However, in cases where the SC is responsible for highly incomplete frames $> 60\%$, the noise can be apparent in some areas (e.g., nonrigid areas, areas with high motion between consecutive frames). To remove these abnormalities, the following fine reconstruction step is utilized.

2.3.1 Initialization Strategy via RPCA

For the denoising of the first $\bar{n}$ frames, we follow a batch approach in order to exploit more effectively their coherence using RPCA. We initially create a spatiotemporal matrix $\mathbf{E} \in \mathbb{R}^{k_f\times 3\bar{n}}$ according to:

$$\mathbf{E} = \begin{bmatrix}\mathbf{n_{c1}}(1) & \mathbf{n_{c1}}(2) & \dots & \mathbf{n_{c1}}(\bar{n})\\ \mathbf{n_{c2}}(1) & \mathbf{n_{c2}}(2) & \dots & \mathbf{n_{c2}}(\bar{n})\\ \vdots & \vdots & \ddots & \vdots\\ \mathbf{n_{c}}_k(1) & \mathbf{n_{c}}_k(2) & \dots & \mathbf{n_{c}}_k(\bar{n})\end{bmatrix} \tag{2.13}$$

where $\bar{n} \ll n$ and $\mathbf{n_c}_i(\bar{n}) = [\mathbf{n_{cx}}_i(\bar{n});\ \mathbf{n_{cy}}_i(\bar{n});\ \mathbf{n_{cz}}_i(\bar{n})]$ represents the i^{th} centroid normal of $\bar{n}^{th}$ frame. The matrix $\mathbf{E}$ may be decomposed as: $\mathbf{E} = \mathbf{S} + \mathbf{N}$ where $\mathbf{S}$ is a low-rank matrix representing the real data while $\mathbf{N}$ is a sparse matrix representing the space where the noise lies. The low-rank matrix $\mathbf{S}$ can be recovered by solving the following convex optimization problem [7]:

$$\text{minimize}\ \ \|\mathbf{S}\|_* + \lambda\|\mathbf{N}\|_1,\ \ \text{subject to}\ \ \mathbf{S}+\mathbf{N}=\mathbf{E} \tag{2.14}$$

where $\|\mathbf{S}\|_*$ denotes the nuclear norm of the matrix which is the sum of the singular values of $\mathbf{S}$. Then we use the elements of the low-rank matrix $\mathbf{S}$ to refine the $\bar{n}$ meshes updating the positions of their vertices.

2.3.2 Online Refining using ISVD

The initialization strategy is applied once (i.e., only for the patch of the first $\bar{n}$ noisy frames), then we use the knowledge of the reconstructed frames in order to estimate the denoised normals of any new presented frame, using an incremental approach. The SVD updating algorithm [11], [6], provides an efficient way to carry out the SVD of a larger matrix $[\mathbf{S}_{k_f\times 3\bar{n}}, \mathbf{B}_{k_f\times 3r}]$, where $\mathbf{B}$ is an $k_f \times 3r$ matrix consisting of the k_f centroid normals of the r additional frames. The matrix $\mathbf{S}$ is an already low-rank matrix consisting of the denoised normals of the $\bar{n}$ previous frames. Specifically, for the normal's estimation of the $\bar{n}+1$ frame, the matrix $\mathbf{S}$ is used, while for any frame $> \bar{n}+2$ the matrix $\mathbf{S}$ is updated as shown in Eq. (2.18). The $r \leq \bar{n}$ represents the number of the observed noisy frames on which we want to estimate the denoised normals. By exploiting the orthonormal properties and block structure, the SVD computation of $[\mathbf{S}, \mathbf{B}]$ can be efficiently carried out by using the smaller matrices, $\mathbf{U}_q, \mathbf{V}_q$, and the SVD of the smaller matrix $\begin{bmatrix}\boldsymbol{\Lambda}_q & \mathbf{U}_q^T\mathbf{B}\\ \mathbf{0} & \mathbf{R}\end{bmatrix}$. Firstly, we apply a qr(.) decomposition of the $(\mathbf{I} - \mathbf{U}_q\mathbf{U}_q^T)\mathbf{B}$ in order to estimate the matrices $\mathbf{Q}$ and $\mathbf{R}$:

$$\mathbf{QR} = \text{qr}((\mathbf{I} - \mathbf{U}_q\mathbf{U}_q^T)\mathbf{B}) \tag{2.15}$$

Next, we obtain the q-rank SVD of the $(q+r)\times(q+r)$ matrix:

$$\begin{bmatrix}\boldsymbol{\Lambda}_q & \mathbf{U}_q^T\mathbf{B}\\ \mathbf{0} & \mathbf{R}\end{bmatrix} = \hat{\mathbf{U}}\hat{\boldsymbol{\Lambda}}\hat{\mathbf{V}}^T \tag{2.16}$$

where r' is the rank of $(\mathbf{I} - \mathbf{U}_q\mathbf{U}_q^T)\mathbf{B}$. Then, the best q-rank approximation of $[\mathbf{S}, \mathbf{B}]$ is:

$$[\mathbf{S}, \mathbf{B}] = ([\mathbf{U}_q, \mathbf{Q}]\hat{\mathbf{U}})\hat{\boldsymbol{\Lambda}}(\begin{bmatrix}\mathbf{V}_q & \mathbf{0}\\ \mathbf{0} & \mathbf{I}\end{bmatrix}\hat{\mathbf{V}})^T \tag{2.17}$$

Finally, the matrix $\mathbf{B}$ obtains the denoised normals of the new frame which will be used to update the vetrices. Matrix $\mathbf{S}$ will be updated, by a left shifting operation denoting as $\mapsto$, in order to obtain the most recent information for more efficient online estimation of the denoised normals of the next frame:

$$\mathbf{S} : (\dot{\mathbf{n}}_{\mathbf{c}1},\ \dot{\mathbf{n}}_{\mathbf{c}2},\ \cdots,\ \dot{\mathbf{n}}_{\mathbf{c}(\bar{n}-1)},\ \dot{\mathbf{n}}_{\mathbf{c}\bar{n}}) \mapsto (\dot{\mathbf{n}}_{\mathbf{c}2},\ \dot{\mathbf{n}}_{\mathbf{c}3},\ \cdots,\ \dot{\mathbf{n}}_{\mathbf{c}\bar{n}},\ \mathbf{B}) \tag{2.18}$$

where $\dot{\mathbf{n}}_{\mathbf{c}i}$ represents the i^{th} row of matrix $\mathbf{S}$.

2.4 Ideal Normals for Vertex Updating

We assume that some points are more reliable than others. Two parameters affect mostly the classification of a point as trustable or not. The first one is (i) the rigidness $\Phi(i)$ of the area (first-ring) where a point lies and the second is (ii) the total distance $\boldsymbol{\beta}_i$ that a point covers through the frames. The criterion that is used for the characterization of a point as rigid or not depends on the percentage change of its first-ring area between two consecutive frames and it is described as:

$$\Phi(i) = \begin{cases}1 & \text{if } \frac{|\text{area}(i,l)-\text{area}(i,l-1)|}{\max(\text{area}(i,l),\text{area}(i,l-1))} > 1\%\\ 0 & \text{otherwise}\end{cases} \tag{2.19}$$

where area(i,l) is the first-ring area of point i as it appears in the l frame and $\Phi(i) = 1$ means that the i point needs to be updated for more accurate results. Regarding to the second parameter, the temporal factor C_t is used, as it has been defined in Eq. (2.5), giving emphasis to the known and less moving points. Additionally, in order to make the process more time efficient, the known points included in the $\mathbf{M}_l$ set, are excluded from the updating process. The fine-tuned normals are then used to update the vertices according to [10]:

$$\mathbf{v}_i^{(e+1)} = \mathbf{v}_i^{(e)} + \frac{\sum_{j\in\boldsymbol{\Psi}_i}\boldsymbol{\beta}_j\bar{\mathbf{n}}_{cj}(\langle\bar{\mathbf{n}}_{cj}, (\mathbf{c}_j^{(e)} - \mathbf{v}_i^{(e)})\rangle)}{|\boldsymbol{\Psi}_i|}\ \forall\ i \notin \mathbf{M}_l,\ \Phi(i) = 1 \tag{2.20}$$

$$\boldsymbol{\beta}_i = \begin{cases}4C_t(i) & \text{if } \mathbf{v}_i \text{ is known}\\ C_t(i) & \text{otherwise}\end{cases} \tag{2.21}$$

$$\mathbf{c}_j^{(e+1)} = (\mathbf{v}_{j1}^{(e+1)} + \mathbf{v}_{j2}^{(e+1)} + \mathbf{v}_{j3}^{(e+1)})/3\ \ \forall\ j \in \boldsymbol{\Psi}_i \tag{2.22}$$

where $\langle\mathbf{a}, \mathbf{b}\rangle$ represents the dot product of $\mathbf{a}$ and $\mathbf{b}$, (e) represents the number of iteration and matrix $\boldsymbol{\Psi}_i$ is the cell of vertices that are directly connected with the vertex $\mathbf{v}_i$.

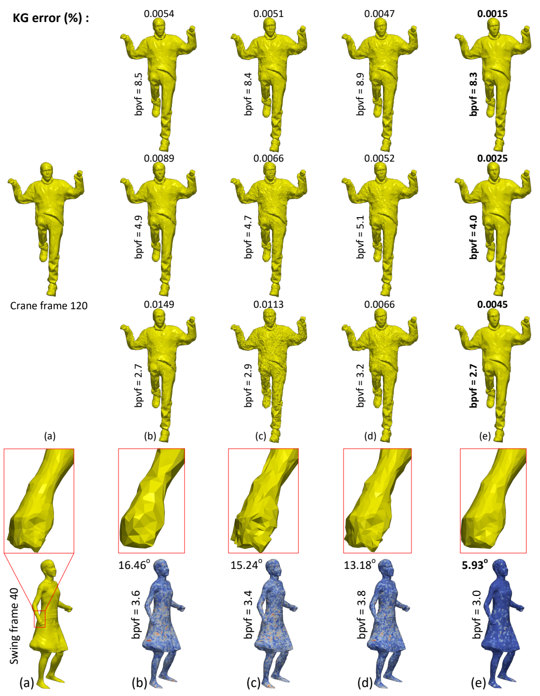


Figure 2: [**Up**] KG error of the reconstructed results, [**Down**] Heatmap visualizing the θ metric per face in different colors. (a) Original mesh and reconstructed using: (b) EFSCA [1], (c) the lifting approach of the FAMC method [8], (d) the DCT approach of the FAMC method [8], (e) our approach.

3 Experimental Analysis and Case Study

The results of the proposed method are compared with: (a) A layer decomposition approach using an Efficient Fine-granular Scalable Coding Algorithm (EFSCA) [1] and (b) The Frame-based Animated Mesh Compression (FAMC) method [8]. To simulate variable bandwidth capabilities and adjust rates for chunks we separate the whole animation in blocks-of-frames (e.g., 10 frames per block) and at any block, a different bpvf is used. In Fig. 2-[Up], we present comparisons between our method and others using different rates of bpvf. For the evaluation we use the KG error metric. In Fig. 2-[Down], we present the heatmap visualization of θ metric for different reconstructed models. Additionally, we provide the mean θ of each frame for the different approaches, as well as enlarged detail of the reconstructed models for easier comparison.

Lung diseases affect the daily routine and the quality of life of many people. Recently, 3D animated simulation of the patient's respiration system has been used by medical experts for a personalized and online observation of their patients' lungs functionality [5, 4], utilizing this approach as a tool for primary diagnosis. Fig. 3 presents an example of a 3D animated model[1] in two different respiratory situations (i.e., exhale and inhale).

4 Conclusions

This work presents an efficient approach for online SC of dynamic 3D meshes. This method is totally parameter-free and it can be used without further changes or extra parameterization. A significant advantage of the proposed method is its ability to transmit different bpv per each frame depending on the instant network's capability. Additionally, the selection of

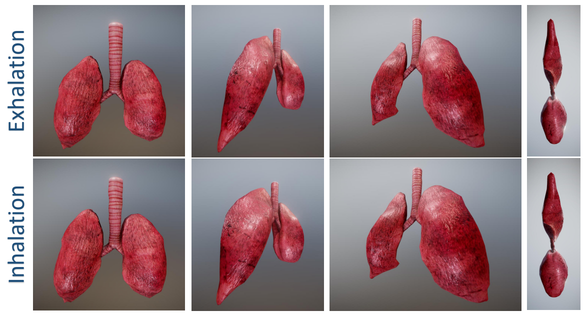


Figure 3: 3D animated lung in different views.

the transmitted vertices is optimized, taking into account both the spatial and temporal information. This gives an extra benefit to the reconstruction process to handle more efficiently the received vertices providing more accurate results.

Acknowledgment

This research has been co-financed by the European Regional Development Fund of the European Union and Greek national funds through the Operational Program Competitiveness, Entrepreneurship and Innovation, under the call RESEARCH - CREATE - INNOVATE (project code:T1EDK-03832).

References

[1] J. K. Ahn, Y. J. Koh, and C. S. Kim. Efficient fine-granular scalable coding of 3d mesh sequences. *IEEE Transactions on Multimedia*, 15(3):485–497, April 2013.

[2] G. Arvanitis, A. S. Lalos, K. Moustakas, and N. Fakotakis. Weighted regularized laplacian interpolation for consolidation of highly-incomplete time varying point clouds. In *2017 3DTV Conference: The True Vision - Capture, Transmission and Display of 3D Video (3DTV-CON)*, pages 1–4, June 2017.

[3] G. Arvanitis, A. Spathis-Papadiotis, A. S. Lalos, K. Moustakas, and N. Fakotakis. Outliers removal and consolidation of dynamic point cloud. In *2018 25th IEEE International Conference on Image Processing (ICIP)*, pages 3888–3892, Oct 2018.

[4] B. Fuerst, T. Mansi, Jianwen Zhang, P. Khurd, J. Declerck, T. Boettger, Nassir Navab, J. Bayouth, Dorin Comaniciu, and A. Kamen. A personalized biomechanical model for respiratory motion prediction. In Nicholas Ayache, Hervé Delingette, Polina Golland, and Kensaku Mori, editors, *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2012*, pages 566–573, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.

[5] Felix G. Hamza-Lup, Anand P. Santhanam, Celina Imielinska, Sanford Meeks, and Jannick P. Rolland. Distributed augmented reality with 3d lung dynamics – a planning tool concept. 2018.

[6] James T. Kwok and Haitao Zhao. Incremental eigen decomposition. In *IN PROC. ICANN*, pages 270–273, 2003.

[7] Zhouchen Lin, Minming Chen, and Yi Ma. The augmented lagrange multiplier method for exact recovery of corrupted low-rank matrices. *CoRR*, abs/1009.5055, 2009.

[8] K. Mamou, T. Zaharia, and F. Preteux. Famc: The mpeg-4 standard for animated mesh compression. In *2008 15th IEEE International Conference on Image Processing*, pages 2676–2679, Oct 2008.

[9] Thom F Oostendorp, Adriaan van Oosterom, and Geertjan Huiskamp. Interpolation on a triangulated 3d surface. *Journal of Computational Physics*, 80(2):331 – 343, 1989.

[10] X. Sun, P. Rosin, R. Martin, and F. Langbein. Fast and effective feature-preserving mesh denoising. *IEEE Transactions on Visualization and Computer Graphics*, 13(5):925–938, Sept 2007.

[11] H. Zha and H. Simon. On updating problems in latent semantic indexing. *SIAM Journal on Scientific Computing*, 21(2):782–791, 1999.

[1] sketchfab

Control of lines of curvature for plate forming in shipbuilding

Masahito Takezawa[1], Kohei Matsuo[1], and Takashi Maekawa[2]

[1] *Core Manufacturing Technology Department, National Maritime Research Institute, Japan*
[2] *Department of Mechanical Engineering, Yokohama National University, Japan*

Abstract

A ship hull consists of complicated doubly curved surfaces particularly at the bow and stern. In shipbuilding, surface plates are formed by the mixture of cold and hot bending, which is typically called line heating. An efficient plate forming method based on the lines of curvature has recently garnered the attention of researchers. However, the method suffers if the lines of curvature are wavy, or pass near the umbilical points. We herein propose novel methods for smoothing the lines of curvature and sweeping out the umbilics from each partitioned plate by deforming the surface within the prescribed deviation from the designed surface based on nonlinear optimization. We demonstrated the effectiveness of our proposed method by applying it to the bow of a bulk carrier, which was provided by a shipbuilding yard.

1 Introduction

In shipbuilding, exterior steel plates are formed using two major steps. First, a flat steel plate is press bent through cold bending to provide a rough bend. Thereafter, the plate is contracted locally using hot bending, a process known as *line heating*, to form the plate into the desired three-dimensional shape.

Lines of curvature are a set of curves on a surface whose tangent at each point is in the principal direction, and possess various interesting geometric characteristics [7]. Many studies on the design and manufacturing of shapes fully utilizing the curvature characteristics have been developed recently [3, 9]. Among them, an efficient method for the forming of ship hull plates based on the lines of curvature was proposed [5, 3] (see Fig. 1). This method allows only expansion to occur along the lines of the curvature when unfolding an input surface onto a plane. Hence, the manufacturer can determine the initial shape of the plate to be cut, specify the pressing direction and angles, and estimate the amount of contraction required to form the shape using line heating.

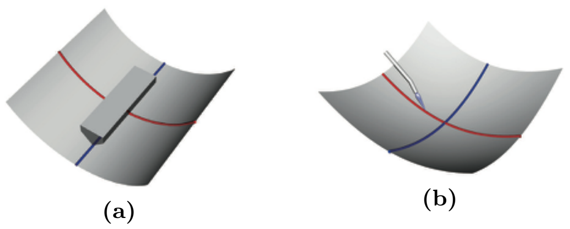


Figure 1: LoC-based plate forming (adapted from [3]): (a) Cold bending by pressing along the line of curvature with a smaller magnitude of curvature (blue line). (b) Line heating using local heat treatment along the line of curvature with a larger magnitude of curvature (red line).

Lines of curvature are generally more sensitive to surface irregularities than the first-order interrogation tools, such as zebra mapping, reflection lines and so on. Therefore, the input surface must be sufficiently fair to obtain a smooth flow of lines of curvature, as shown in Fig. 2 (a). In addition, the existence of umbilics, which are the singularities of the lines of curvature, destroys the nice orthogonal net of the lines of curvature and induces sharp turns, as illustrated in Fig. 2 (b). These two facts render the application of lines-of-curvature (LOC)-based pressing and line heating for plate forming difficult to achieve.

We herein propose novel methods for removing these two obstacles when forming plates through LoC-based pressing and line heating. The primary contributions of this paper are as follows:

- We propose an interactive method for smoothing the principal direction fields within the parameter space using Gaussian filtering as a preprocessing step for smoothing the lines of curvature.
- We modify the input B-spline surface based on the smoothed principal direction fields based on optimization.
- We sweep out the umbilical point from the surface boundary such that the umbilics are not inside the plate.

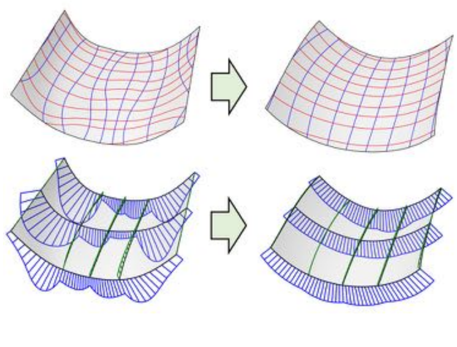
2 Related work

2.1 LoC-based plate forming

Matsuo and Matsuoka [5] introduced a plate forming-method based on the lines of curvature of the plate surfaces. The key idea is that the surface is bent the most along the lines of curvature; hence, they indicate the directions for bending. The authors flatten doubly curved plates onto a plane by utilizing the fact that the geodesic curvature along a line of curvature exhibits the same curvature as that of a flattened curve. In other words, a line of curvature can be considered as an edge of a developable surface formed using a vector orthogonal to both the surface normal and tangent of the line of curvature. Accordingly, it can be developed onto a plane based on the fact that curves on isometric surfaces exhibit the same geodesic curvature at the corresponding points [3].

2.2 Fairing

To apply LoC-based plate forming, the input surface must exhibit a smooth flow of lines of curvature. Numerous studies have been conducted on the smoothing of B-spline surfaces. Two typical approaches exist for creating fair B-spline surfaces [2]: (1) fitting surfaces with fairing terms (e.g., [1]) and (2) post-processing (e.g., [2, 4]). Our fairing method belongs to the second category. Various post-processing surface fairing methods have been described in the literature. For example, Kawasaki et al. [4] proposed a method for fairing surfaces while maintaining the characteristic shapes, such as sharp edges, based on a normal map image of the surface. This approach is a first-order interrogation tool.



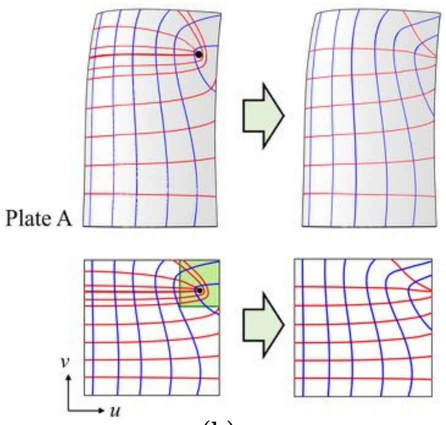


Figure 2: Two issues that render the applications of LoC-based plate forming difficult: (a) smoothing the flow of lines of curvature (upper images), where curvature plots along iso-parametric curves are smoothed as a byproduct (lower images), and (b) sweeping out an umbilical point indicated by the black point where the upper images are in 3D geometry and the lower images are the pre-images (model: plate A).

In the present study, we introduce a method for smoothing the flow of curvature lines by smoothing the principal direction fields within the parameter space.

3 Control of lines of curvature based on principal direction fields

3.1 Smoothing of principal direction fields

At each non-umbilical point, two principal directions exist that are orthogonal; hence, the lines of curvature form an orthogonal net of lines. However, the corresponding two principal directions in the parameter space are generally not orthogonal. We evaluate the principal directions at each uniform grid point within the uv-parameter space of the B-spline surface.

According to [11], the principal directions in the parameter space can be categorized as "two independent pairs of directions with π rotationally symmetric within each pair." Therefore, each principal direction generates an independent *line field*. The angle of the line in parameter space ψ with respect to the u axis is defined within the range $-\frac{\pi}{2} < \psi \leq \frac{\pi}{2}$.

We subsequently apply Gaussian filtering interactively to the regions where the principal direction fields are not smooth. Gaussian filtering is a technique used in image processing to smooth images and remove noises [10]. Let us denote the angle of the principal direction at the grid point $\mathbf{p}$ in the uv space as $\psi_{\mathbf{p}}$, and at its neighboring points $\mathbf{q} \in N(\mathbf{p})$ as $\psi_{\mathbf{q}}$. The angle of the principal direction is updated iteratively using Gaussian filtering as follows:

$$\psi_{\mathbf{p}}^{(k+1)} = \psi_{\mathbf{p}}^{(k)} + \Delta\psi_{\mathbf{p}}^{(k)} , \tag{3.1}$$

where the superscript (k) denotes the k-th iteration, and $\Delta\psi_{\mathbf{p}}^{(k)}$ is defined as

$$\Delta\psi_{\mathbf{p}}^{(k)} = \frac{\sum_{\mathbf{q}\in N(\mathbf{p})} W(||\mathbf{p}-\mathbf{q}||)(\psi_{\mathbf{q}}^{(k)} - \psi_{\mathbf{p}}^{(k)})}{\sum_{\mathbf{q}\in N(\mathbf{p})} W(||\mathbf{p}-\mathbf{q}||)} , \tag{3.2}$$

where $-\frac{\pi}{2} \leq \psi_{\mathbf{q}}^{(k)} - \psi_{\mathbf{p}}^{(k)} \leq \frac{\pi}{2}$ and $||\mathbf{x}||$ denotes the Euclidean norm. The weight $W(d) = e^{-d^2/2\sigma^2}$ is the Gaussian distribution function with σ being the standard deviation of the Gaussian distribution determined by the user. The filter size in the principal direction evaluation points, which is the neighborhood considered for filtering, is determined by the user.

We apply a Gaussian filter to either the maximum or minimum principal direction. If one of the directions is filtered, the other direction is determined automatically from the property in which the two principal directions are orthogonal to each other within the geometry space. In this study, we chose the minimum principal direction, although the maximum principal direction can be applied without affecting the outcome.

3.2 Visualization of smoothed principal direction fields

We discuss the smoothing of the principal direction fields such that the lines of curvature, which are integral curves for the principal direction fields, become smooth. We introduce the so-called *approximated LoC* in the parameter space to verify the smoothness of the principal direction fields. Tracing the *approximated LoC* is given as the initial value problem, using standard numerical techniques such as the Runge-Kutta method. The angle in uv space is interpolated by smoothed principal directions.

3.3 Reflecting modified principal direction fields onto the input surface

We consider the energy functional F, which consists of a linear combination of three separate energy functions, to optimize the shape of the surface based on the principal direction fields. Let $\mathbf{R}$=$(r_1, r_2, ..., r_N)$ be the selected components of the control points of the input B-spline surface that will be modified through the smoothing process. The energy functional F is given as follows:

$$F(r_1, r_2, ..., r_N) = F_d + w_p F_p + w_f F_f \tag{3.3}$$

where w_p and w_f are the weighting factors for each energy term defined by the user. The first term ensures that the new control points $\mathbf{P}_{ij}$ do not deviate significantly from the original control points $\bar{\mathbf{P}}_{ij}$:

$$F_d = \sum_{i=0}^{m}\sum_{j=0}^{n} ||\mathbf{P}_{ij} - \bar{\mathbf{P}}_{ij}||^2 , \tag{3.4}$$

where $(m+1)$ and $(n+1)$ are the number of control points in the u and v directions, respectively. The second term ensures that the principal directions in the parameter space, ψ_{max} and ψ_{min} of the optimized surface, match those of the smoothed surfaces, $\bar{\psi}_{max}$ and $\bar{\psi}_{min}$, as much as possible:

$$F_p = \sum_{i=0}^{M} (\psi_{max\ i} - \bar{\psi}_{max\ i})^2 + (\psi_{min\ i} - \bar{\psi}_{min\ i})^2 \ , \qquad (3.5)$$

$-\frac{\pi}{2} \le \psi_{max\ i} - \bar{\psi}_{max\ i} \le \frac{\pi}{2}$ and $-\frac{\pi}{2} \le \psi_{min\ i} - \bar{\psi}_{min\ i} \le \frac{\pi}{2}$, and $(M+1)$ is the number of sampling points. The third term F_f is defined as

$$F_f = \int \left(\frac{d\kappa_{max}}{d\mathbf{e}_{max}} \right)^2 + \left(\frac{d\kappa_{min}}{d\mathbf{e}_{min}} \right)^2 dA \ , \qquad (3.6)$$

which minimizes the variation in curvature rather than its magnitude; thus, the resulting surface is referred to as the minimum variation surface [6]. Eq. (3.6) evaluates the area integral of the sum of the squared magnitudes of the derivatives of the normal curvatures from the corresponding smoothed principal directions. The unit vectors $\mathbf{e}_{max}$ and $\mathbf{e}_{min}$ are the smoothed maximum and minimum principal directions at the evaluated points, respectively.

To minimize the objective function in Eq. (3.3), we used the *bound optimization by quadratic approximation* (BOBYQA) method ([8] of the Numerical Algorithms Group program, NAG Mark26.1 CLW32261EL e04jcc).

3.4 Sweeping out of umbilics from surface boundaries

If umbilical points exist near the four edges of the plate, our strategy is to sweep them out from the nearest boundary using the techniques described in Section 3.1.

4 Results

We apply our algorithms to the bow surface of a bulk carrier of length 278 m, which is provided by a shipbuilding yard.

4.1 Smoothing of curvature lines

The upper and lower rows of Fig. 3 depict the curvature lines before and after the application of smoothing, respectively. If we use a large σ of the Gaussian function to smooth the entire domain, the flow of the lines of curvature may change significantly. Accordingly, we applied Gaussian filtering through three steps.

As shown in the lower row of Fig. 3, the flow of the curvature lines becomes smoother and the zebra mapping achieves a higher quality. Using our method, we improved the fairness of the hull surface and smoothed the flow of the curvature lines simultaneously. Based on practical experience, we observed that the choice of weights w_p and w_f affects the surface deformation significantly. To maintain the deformation within 0.05% of the ship length, we used $w_p = 5.0$ and $w_f = 2.0$ for the computation of the bow surface. The maximum deformation is 101 mm, which is within the allowable deformation. The computational time for the bow surface model required approximately 40 min owing to the relatively large number of variables used in the optimization, namely, 378.

4.2 Sweeping out of umbilics from surface boundaries

Three umbilics were detected in the bow surface. The bow surface was partitioned into 28 manufacturable quadrilateral plates such that the three umbilics were located near the plate boundaries (see Fig. 4). Because plate surfaces A, B, and C are trimmed surfaces, we fit an accurate plate surface from the dense points extracted from each trimmed surface. We subsequently applied Gaussian filtering to the boundary regions of the plate surface where the umbilical points exist to sweep them out. The computational results of surface optimization are illustrated in Figs. 2 (b) and 5. The maximum distance deviation from the trimmed surface caused by the sweeping out of the umbilics was 1.0 mm or less in all three cases, which is negligible considering that the hull plate is formed manually by craftsmen.

5 Conclusion

We introduced a method for smoothing the lines of curvature through smoothing the principal direction fields. Furthermore, we provided a method to sweep out the umbilics from the partitioned quadrilateral plates such that the proposed LoC-based plate-forming method can be applied practically to actual shipbuilding. Although we applied these techniques to shipbuilding, they can be applied easily to other engineering fields.

The limitations of the proposed method are as follows:

- If umbilical points exist stably at the center of the surface, such points cannot be swept out to the surface boundary under a small amount of deformation.
- When an input principal direction field cannot be realized geometrically, such as when the boundary curves are restrained, the surface cannot be corrected into a shape satisfying the input principal direction field.

In the future, we plan to form ship plates in a shipyard based on the developments of our proposed method.

References

[1] Ulrich Dietz. Fair surface reconstruction from point clouds. In *Proceedings of the International Conference on Mathematical Methods for Curves and Surfaces II Lillehammer, 1997*, pages 79–86, Nashville, TN, USA, 1998. Vanderbilt University.

[2] Stefanie Hahmann and Stefan Konz. Fairing bi-cubic B-spline surfaces using simulated annealing. *Curves and Surfaces with Applications in CAGD*, pages pp. 159–168, 1997.

[3] H. Joo, T. Yazaki, M. Takezawa, and T. Maekawa. Differential geometry properties of lines of curvature of parametric surfaces and their visualization. *Graphical Models*, 76(4):pp. 224–238, 2014.

[4] Taro Kawasaki, Pradeep Kumar Jayaraman, Kentaro Shida, Jianmin Zheng, and Takashi Maekawa. An image processing approach to feature-preserving B-spline surface fairing. *Computer-Aided Design*, 99:pp. 1–10, 2018.

[5] K. Matsuo and K. Matsuoka. Development of new system for developing curved shell plates of ships. *Transactions of the Japan Society of Mechanical Engineers. C*, 76(771):pp. 2797–2802, 2010. In Japanese.

[6] Henry P. Moreton and Carlo H. Séquin. Functional optimization for fair surface design. *SIGGRAPH Comput. Graph.*, 26(2):pp. 167–176, July 1992.

[7] N. M. Patrikalakis and T. Maekawa. *Shape Interrogation for Computer Aided Design and Manufacturing.* Springer-Verlag, Heidelberg, 2002.

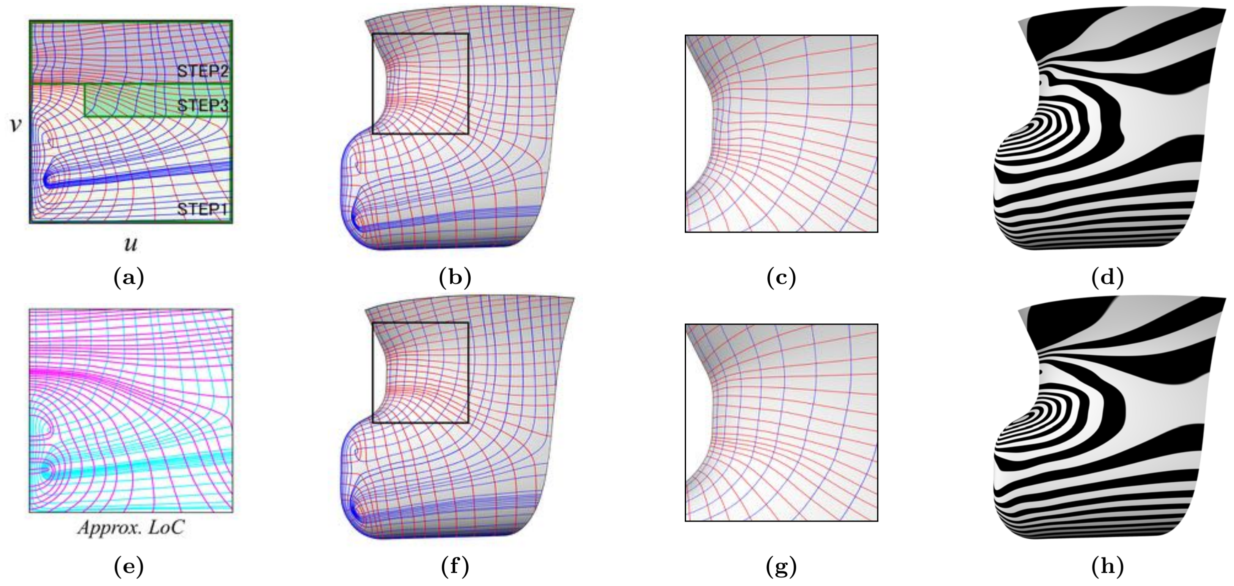


Figure 3: Bow surface optimization: Upper row, before optimization; lower row, after optimization. Left-to-right: Lines of curvature in the parameter space, lines of curvature in the geometry space, close-up view, and zebra mapping.

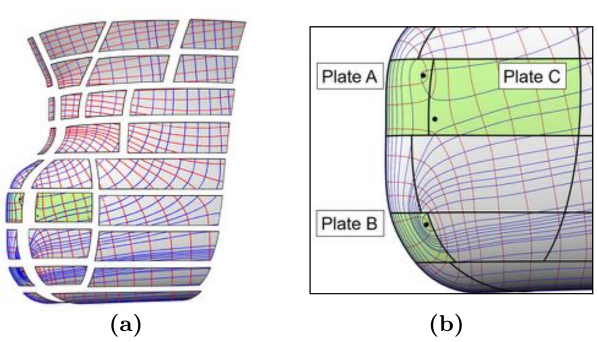


Figure 4: Partitioning of bow surface: (a) partitioned into 28 plates, (b) close-up view of (a), where the black points indicate umbilical points.

[8] Michael JD Powell. The BOBYQA algorithm for bound constrained optimization without derivatives. *Cambridge NA Report NA2009/06, University of Cambridge, Cambridge*, pages pp. 26–46, 2009.

[9] M. Takezawa, T. Imai, K. Shida, and T. Maekawa. Fabrication of freeform objects by principal strips. *ACM Transactions on Graphics (TOG)*, 35(6):225:1–225:12, 2016.

[10] Carlo Tomasi and Roberto Manduchi. Bilateral filtering for gray and color images. In *Computer Vision, 1998. Sixth International Conference on*, pages 839–846. IEEE, 1998.

[11] Amir Vaxman, Marcel Campen, Olga Diamanti, David Bommes, Klaus Hildebrandt, Mirela Ben-Chen Technion, and Daniele Panozzo. Directional field synthesis, design, and processing. In *ACM SIGGRAPH 2017 Courses*, SIGGRAPH '17, pages 12:1–12:30, 2017.

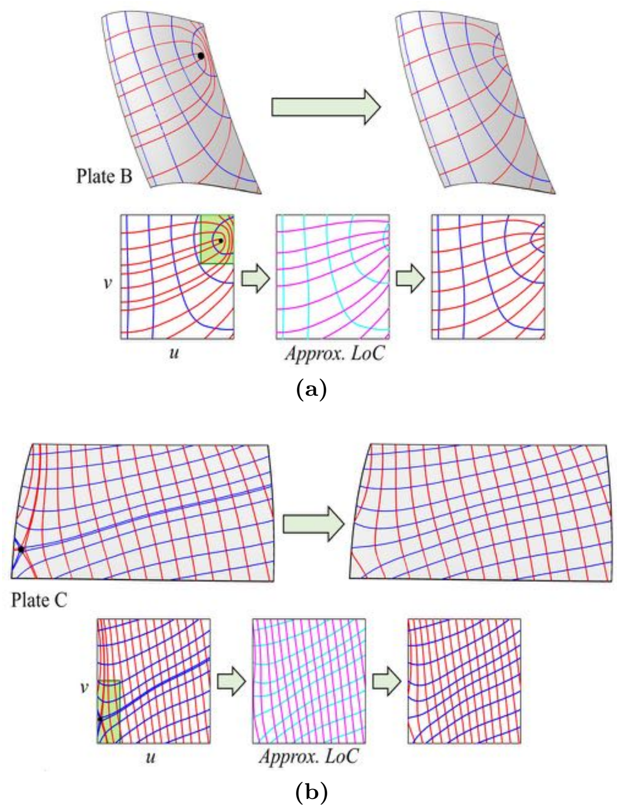


Figure 5: Top to bottom: plate surface with lines of curvature in the geometry and parameter spaces, in that order. Left to right: before and after shape optimization. Magenta and cyan indicate the *approximated LoC* of the input principal direction fields. Plates (a) B and (b) C.

Detection of discontinuities from scattered data

Cesare Bracco [1], Oleg Davydov [2], Carlotta Giannelli [1], and Alessandra Sestini [1]

[1] *Department of Mathematics and Computer Science "U. Dini" - University of Florence*
[2] *Department of Mathematics - Justus Liebig University Giessen*
Email: cesare.bracco@unifi.it, oleg.davydov@math.uni-giessen.de, carlotta.giannelli@unifi, alessandra.sestini@unifi.it

Abstract

We present a method for detecting from scattered data discontinuities and gradient discontinuities, that is, faults and gradient faults, respectively. Our approach consists of using new fault indicators which allow to detect clouds of points enclosing the faults, which can be used to reconstruct the shape of the fault curves. The fault indicators are based on recently introduced minimal numerical differentiation formulas for gradient or Laplacian on irregular centers, which eliminates the need for any intermediate gridding of the data. The choice of our indicators is motivated by their theoretical properties: their asymptotic behavior when the spacing between the data sites goes to zero is related to the presence of discontinuities. A selection of numerical examples illustrates the performance of the method and highlights its potential range of application.

1 Discontinuity curves: faults and gradient faults

Let $\Omega \subset \mathbb{R}^2$ and let f be a smooth function in $\Omega \setminus (\mathcal{F}_O \cup \mathcal{F}_G)$, where both $\mathcal{F}_O$ and $\mathcal{F}_G$ are unions of curves such that f is discontinuous at points $x \in \mathcal{F}_O$ and its gradient is discontinuous at $x \in \mathcal{F}_G$. We call the curves composing $\mathcal{F}_O$ and $\mathcal{F}_G$ *ordinary faults* and *gradient faults*, respectively.

Given a set of scattered points $X \subset \Omega$ with associated function values $f(\mathbf{x})$ for each $\mathbf{x} \in X$, we are interested in detecting the ordinary and gradient faults of f. This problem has been extensively investigated in the literature (see, e.g., [1, 2, 4, 5, 8]).

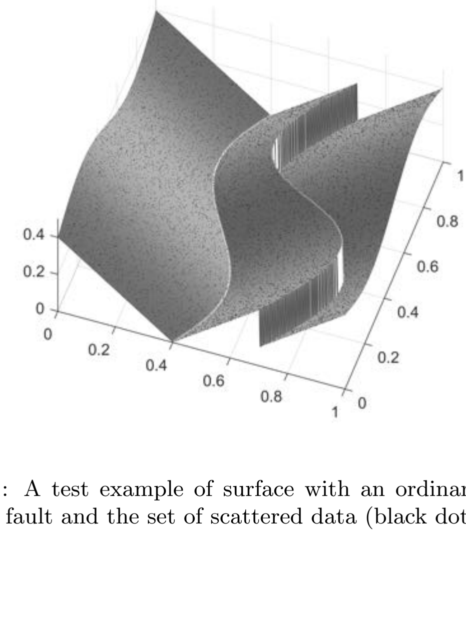


Figure 1: A test example of surface with an ordinary and a gradient fault and the set of scattered data (black dots).

2 Fault detection

We define two indicators, I_∇ and I_Δ:

$$I_\nabla(\mathbf{x}, N_\mathbf{x}^\nabla) := \frac{\| \sum_{\mathbf{x}_j \in N_\mathbf{x}^\nabla} \bar{\bar{\mathbf{w}}}_j f(\mathbf{x}_j)\|_2}{\big\| \sum_{\mathbf{x}_j \in N_\mathbf{x}^\nabla} |\bar{\bar{\mathbf{w}}}_j| \|\mathbf{x}_j - \mathbf{x}\|_2 \big\|_2}, \tag{2.1}$$

$$I_\Delta(\mathbf{x}, N_\mathbf{x}^\Delta) := \frac{| \sum_{\mathbf{x}_j \in N_\mathbf{x}^\Delta} \bar{\bar{w}}_j f(\mathbf{x}_j)|}{\sum_{\mathbf{x}_j \in N_\mathbf{x}^\Delta} |\bar{\bar{w}}_j| \|\mathbf{x}_j - \mathbf{x}\|_2^2}, \tag{2.2}$$

where $N_\mathbf{x}^\nabla, N_\mathbf{x}^\Delta \subset X$ are local subsets including $\mathbf{x}$ and the $\bar{\bar{\mathbf{w}}}_j$ and $\bar{\bar{w}}_j$ are the weights of minimal numerical differentiation formulas [6] for gradient and Laplacian operators, respectively.

Theorem 1 *The indicator $I_\nabla(\mathbf{x}, N_\mathbf{x}^\nabla)$ ($I_\Delta(\mathbf{x}, N_\mathbf{x}^\Delta)$) is bounded for $N_\mathbf{x}^\nabla \cap \mathcal{F}_O = \emptyset$ ($N_\mathbf{x}^\Delta \cap (\mathcal{F}_O \cup \mathcal{F}_G) = \emptyset$).*

Theorem 2 *Let $\{\mathbf{x}_n \in \Omega\}_{n=1,\dots,\infty}$ such that $\lim_{n\to\infty} \mathbf{x}_n = \mathbf{x}_O \in \mathcal{F}_O$. If the corresponding $N_{\mathbf{x}_n}^\nabla$ are local subsets of X sharing the same geometry, with diameter converging to 0 and containing points on both sides of an ordinary fault, then*

$$\lim_{n\to\infty} I_\nabla(\mathbf{x}_n, N_{\mathbf{x}_n}^\nabla) = \infty.$$

If $\lim_{n\to\infty} \mathbf{x}_n = \mathbf{x}_F \in \mathcal{F}_O \cup \mathcal{F}_G$, analogous assumptions on the $N_{\mathbf{x}_n}^\Delta$ imply

$$\lim_{n\to\infty} I_\Delta(\mathbf{x}_n, N_{\mathbf{x}_n}^\Delta) = \infty.$$

More details (about ordinary fault detection) can be found available in [3].

These two theorems suggest the following detection criteria:

- mark $\mathbf{x} \in X$ as close to a fault if $I_\Delta(x) > \alpha^G$,
- mark $\mathbf{x} \in X$ as close to an ordinary fault if $I_\Delta(x) > \alpha^G$ and $I_\Delta(x) > \alpha^O$,
- mark $\mathbf{x} \in X$ as close to a gradient fault if $I_\Delta(x) > \alpha^G$ and $I_\Delta(x) \le \alpha^O$,

where $\alpha^G = C^G \cdot \mathrm{median}(\{I_\Delta(\mathbf{x}) : \mathbf{x} \in X\})$ and $\alpha^O = C^O \cdot \mathrm{median}(\{I_\nabla(\mathbf{x}) : \mathbf{x} \in X\})$, for suitably chosen constants C^G and C^O. In the case of surfaces with large (almost) flat areas, the medians are very close to 0, and so all points in non-flat areas are detected. Applying the same detection criteria twice allows to properly find the real fault points.

Figure 2 shows the detection of fault points from the data of Figure 1. The red and blue colors correspond to ordinary and gradient fault, respectively.

If necessary, the resulting cloud of points can be employed to reconstruct curves approximating the faults. In Figure an example of fault reconstruction from the detected data of Figure 2 is shown: we used an approach based on the computation of local least squares approximations (see [9, 10]) to "narrow" the point clouds, and then we approximated the obtained points with a suitable cubic spline.

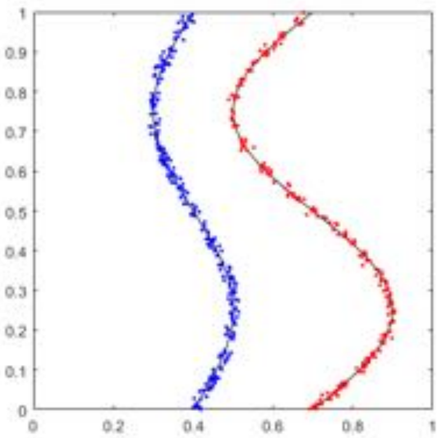


Figure 2: Detected ordinary fault (red) and gradient fault (blue) points from the set of scattered data of Figure 1, compared with the exact fault curves (black lines).

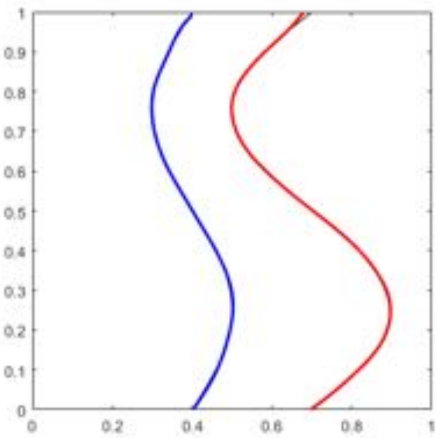


Figure 3: Final reconstruction of the two faults (blue and red lines) from the detected points of Figure 2, compared with the exact fault curves (black lines).

This approach can be also generalized to surfaces without a parametrization of type $S(\mathbf{x}) = (\mathbf{x}, f(\mathbf{x}))$. Let the data be a set of points $Z \subset \mathbb{R}^3$ belonging to a surface S which is G^2 except for a finite number of curves, where it is G^0. We assume that for each $\mathbf{z}$ there exist a plane $\pi_{\mathbf{z}}$ and $f_{\mathbf{z}} : \pi_{\mathbf{z}} \to \mathbb{R}$ such that in a neighbourhood of z the surface S can be seen as the graph of $f_{\mathbf{z}}$. The curves along which S is G^0 correspond to the gradient faults of the functions $f_{\mathbf{z}}$, and then we can locally use the indicator I_Δ to detect them. In practice, $\pi_{\mathbf{z}}$ is determined by principal component analysis on a suitable local subset of data $N_{\mathbf{z}}$, while $f_z(\mathbf{x})$ is the length of the segment joining $\mathbf{x}$ and the intersection between S and the normal to $\pi_{\mathbf{z}}$ through $\mathbf{x}$, for any $\mathbf{x} \in \pi_{\mathbf{z}}$.

3 Applications

The detection of discontinuities is a common problem arising in several applications, ranging from 2D and 3D edge detection in computer graphics to the automatic extraction of features from geophysical data. We present two examples highlighting the potential of our method in these applications.

3.1 Faults in geophysics

Rift areas are characterized by the extension of the crust and of the litoshpere, and they typically show the presence of faults and fractures (see, e.g., [7]). These features can be considered as ordinary faults, which our method is able to detect (see Figure 5).

Note that in some areas, the detected points may determine a "fault area" rather than a "fault curve", which corresponds to a wide fracture. For this reason, no fault curve is reconstructed in this case, since it would mean losing information about the width of the fault area.

3.2 Edge detection

Extracting sharp features, and in particular detecting edges, is a very important task in the context of 3D surface reconstruction for reverse engineering, industrial design and prototyping (see, e.g., [11, 12]). From a mathematical point of view, this problem corresponds to detecting the curves along which a continuous surface is only G^0, which makes it a natural application for our method. In Figure 6 we show the result of the application of our method to the well-known fandisk benchmark model.

4 Conclusions

We defined a fault and gradient fault detection method based on indicators derived from minimal differentiation formulas for scattered data. The algorithm is supported by the theoretical analysis of the indicators' behaviour, which can be generalized and applied to a wide range of topics, including geophysics and engineering and computer graphics. The indicators can be also potentially extended, by using higher-order minimal differential formulas, to detect higher-order discontinuities.

References

[1] G. Allasia, R. Besenghi, and R. Cavoretto. Accurate approximation of unknown fault lines from scattered data. *Acc. Sc. Torino Memorie Sc. Fis.*, 33:1–24, 2009.

[2] M. Bozzini and M. Rossini. The detection and recovery of discontinuity curves from scattered data. *Journal of Computational and Applied Mathematics*, 240:148–162, 2013.

[3] C. Bracco, O. Davydov, C. Giannelli, and A. Sestini. An application of numerical differentiation formulas to discontinuity curve detection from irregularly sampled data. *Rend. Semin. Mat. Univ. Politec. Torino*, 2019. in press.

[4] D. Cates and A. Gelb. Detecting derivative discontinuity locations in piecewise continuous functions from Fourier spectral data. *Numer. Algor.*, 46:59–84, 2007.

[5] A. Crampton and J. C. Mason. Detecting and approximating fault lines from randomly scattered data. *Numer. Algor.*, 39:115–130, 2005.

[6] Oleg Davydov and Robert Schaback. Error bounds for kernel-based numerical differentiation. *Numer. Math.*, 132(2):243–269, 2016.

[7] A. De Matteo, G. Corti, B. van Wyk de Vries, B. Massa, and G. Mussetti. Fault-volcano interactions with broadly

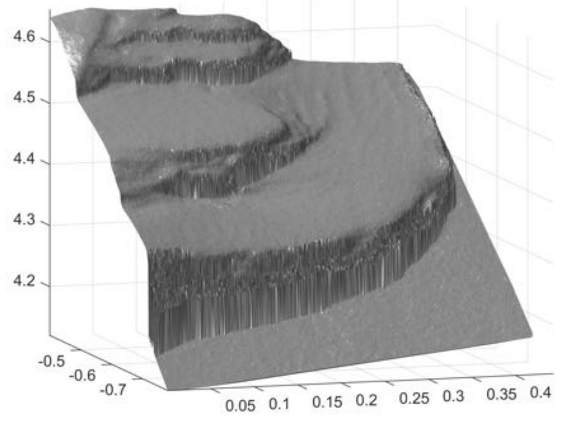


Figure 4: An experimental model simulating a rift area (data courtesy of Giacomo Corti - Istituto di Geoscienze e Georisorse CNR, Florence - Italy).

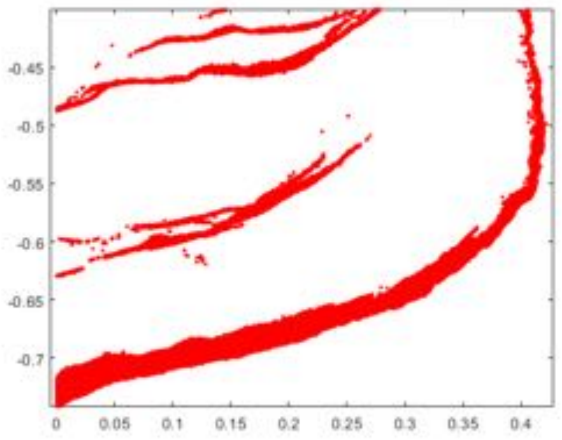


Figure 5: Map of the detected ordinary fault points obtained from the data shown in Figure 4.

distributed stretching in rifts. *J. Volcan. Geoth. Res.*, 362:64–75, 2018.

[8] T. Gutzmer and A. Iske. Detection of discontinuities in scattered data approximation. *Numer. Algorithms*, 16:155–170, 1997.

[9] I. K. Lee. Curve reconstruction from unorganized points. *Comput. Aided Geom. D.*, 17:161–177, 2000.

[10] B. Sober and D. Levin. Manifold approximation by moving least-squares projection (MMLS). *Technical report available on arXiv.org (arXiv:1606.07104v4)*, 2018.

[11] T.-T. Tran, V.-T. Cao, V.-T. Nguyen, Ali S., and D. Laurendeau. Automatic method for sharp feature extraction from 3D data of man-made objects. In *2014 International Conference on Computer Graphics Theory and Applications (GRAPP)*, 2015.

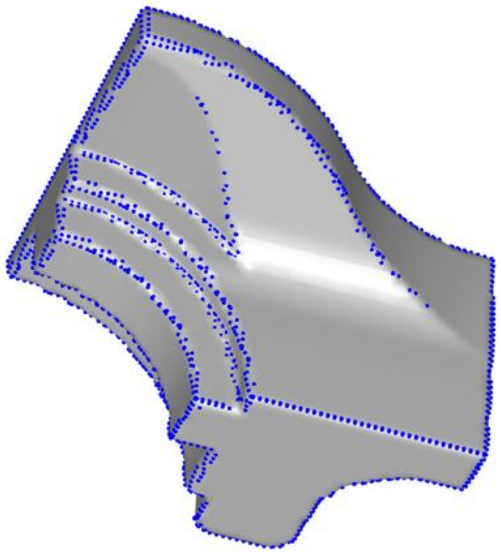


Figure 6: Example of edge detection: surface approximating the "fandisk"benchmark of scattered data (grey surface) and detected points (blue points).

[12] C. Weber, Hahmann, S., and H. Hagen. Sharp Feature Detection in Point Clouds. In *SMI 2010 - Shape Modeling International Conference, Jun 2010, Aix-en-Provence, France*, pages 175–186, 2010.

EdgeNet: Deep Metric Learning for 3D Shapes

Mingjia Chen [1], Qianfang Zou [2], Changbo Wang [1], and Ligang Liu [2]

[1] *School of Computer Science and Software Engineering, East China Normal University, China*
[2] *School of Mathematical Sciences, University of Science and Technology of China, China*

Abstract

We introduce EdgeNet, a metric learning architecture for extracting semantic local shape features, directly applicable to a wide range of shape analysis applications such as point matching, object classification, shape segmentation, and partial registration. EdgeNet is based on a novel technique to keep edge-wise correspondences in the deep feature space and encodes the local structure into the learned features. It is trained under the supervision of edge-wise correspondences by using the 3D coordinates. The training loss combines a bi-triplet loss to enforce feature variations between the semantic matching points in the feature space, a transformation loss to encourage consistency between corresponding edges after alignment transformation, and a smoothness loss guarantees the flatness between the nearest points in the feature space. The learned features are proved to encode local content, structure, and asymmetry for 3D shapes. Our network can be adapted to either 3D meshes or point clouds. We compare the performance of the EdgeNet with existing state-of-the-art approaches and demonstrate the efficiency and efficacy of EdgeNet in three shape analysis tasks, including shape segmentation, partial matching, and shape retrieval.

1 INTRODUCTION

Shape features or descriptors, which map a shape, either locally or globally, into multi-dimensional vectors, play important roles in understanding and analyzing 3D models in various high-level tasks in geometry processing, such as shape correspondence [9], labeling [6], and retrieval [2]. Typically, shape features should be *discriminative* (capturing distinctive attributes), *robust* (invariant with respect to parts missing or noise), *asymmetric* (sensitive to reflective symmetry), and *computationally-efficient*.

We found that two adjacent patches should have similar features in the deep feature space since they probably belong to the same semantic parts. Based on these observations, our key concept is to maintain edge-wise correspondences in feature space and also encode local spatial relations in the learned features, thus resulting in more robust and discriminative features. To this end, we design a novel deep metric learning architecture, called *EdgeNet*, which seeks to preserve feature similarities and spatial relations for all edge pairs in the learned feature space. Figure 1 shows several experimental results.

Contributions. The contributions of our method are summarized in the following:

- A new metric learning architecture, EdgeNet, is proposed for structure-aware local feature learning. It can be adapted for 3D meshes or point clouds.
- An end-to-end feature extractor, PatchNet, is designed to compute local context-, and global position-aware features.

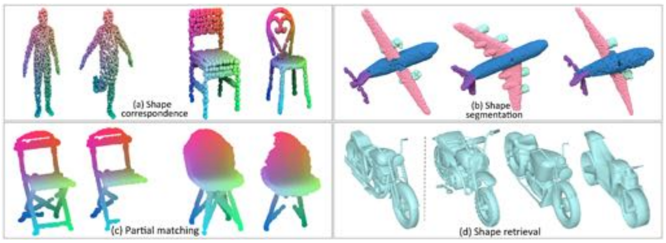


Figure 1: We propose a deep metric learning architecture which learns local shape features in an end-to-end setting. Our deep features can be used in a wide range of shape analysis applications, including (a) shape correspondence, (b) shape segmentation, (c) partial matching, (d) shape retrieval. Corresponding points in (a) and (c) or parts in (b) have the same RGB color.

- A novel context-, structure-, and asymmetry-aware local feature is learned for 3D shapes, which can be directly used in a variety of shape understanding applications.

2 OVERVIEW

We will elaborate our EdgeNet (Figure 2) in detail. It encodes the relative relationships into the learned features under the supervision of edge-wise correspondences.

Input dataset. The input to our method is 3D shapes, which could be either rigid, such as airplane, chair, or motorbike from the corresponding benchmark [5], or non-rigid human FAUST dataset [1]. For each category in the dataset, all shapes are normalized to unit biharmonic distance diameter and have same number of uniformly sampled vertices in correspondence.

Problem. Our goal is to find a mapping f that takes as input any vertex p of a 3D shape and computes a feature (descriptor) $D_p \in \mathbb{R}^d$ in some d dimensional space for that point (we set $d = 128$ in our implementation). The function is designed such that features of corresponding points across shapes are as close as possible to each other in $\mathbb{R}^d$ and features of non-corresponding points across shapes are away over a minimal constant gap to each other.

Local patches. We directly use the (x, y, z) coordinates of each vertex, according to the frame of the bounding box of the shape as shown in Figure 4, as its channels.

For each vertex p_i on a 3D shape, we learn local feature from its neighborhood points instead of itself only. Thus, we define a local patch $P_i = \{p^i_j, j = 0, 1, ..., N\}$ with N points ($N = 120$ by default) where the N points are uniformly sampled from the surface region around p_i within a biharmonic distance r ($r = 0.025$ by default), as shown in Figure 4. The coordinates of vertices in P_i are translated into the local frame centered at p_i.

In order to encode the global information, we use the coordinates of p_i as the global location of P_i.

PatchNet. We design an end-to-end deep network as feature extractor, called *PatchNet*, to map p_i into its feature vector in $\mathbb{R}^d$, as shown in Figure 3.

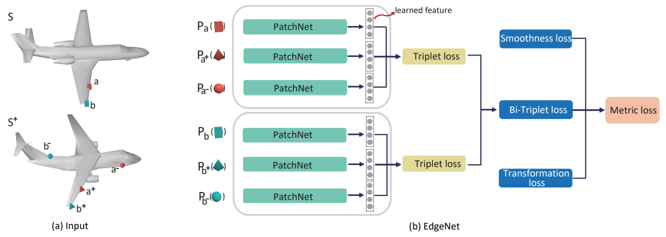


Figure 2: EdgeNet architecture. EdgeNet is composed of two triplet network branches. Each triplet consists of three PatchNets (Figure 3). The six PatchNet share the same parameters. (a) Local patch of each point is the input of PatchNet. Here superscript '+' denotes a corresponding point and '-' denotes a non-corresponding point. There is an edge between a and b, and similarly to a^+ and b^+. (b) EdgeNet is trained by using the combination of the bi-triplet loss, the smoothness loss and the transformation loss.

It is consisted of two sub-networks. The first sub-network, called *local patch encoder* (in green), takes as input the coordinates of all points in P_i and extracts a feature vector in $\mathbb{R}^{d/2}$ encoding the local context of p_i. The second sub-network, called *global location encoder* (in blue), takes as input the coordinates of p_i and extracts a feature vector in $\mathbb{R}^{d/2}$ encoding the global information of p_i. Then we concatenate the two representations generated by the sub-networks as the feature vector $D_p \in \mathbb{R}^d$ of p_i.

All PatchNets for each vertex p_i share the same weight parameters, which will be trained by our EdgeNet.

EdgeNet. The architecture of our EdgeNet is depicted in Figure 2. EdgeNet takes as input a pair of corresponding edges (a, b) and (a^+, b^+) from two shapes S and S^+ respectively. We randomly sample two points a^- and b^- on S^+ and create two triplets (a, a^+, a^-) and (b, b^+, b^-). Here superscript '+' denotes a corresponding point and '-' denotes a non-corresponding point. There is an edge between a and b, and similarly to a^+ and b^+.

For the triplet (a, a^+, a^-) , we use the triplet loss function [15] enforcing that the features of a and a^+ is as close as possible and the feature of a and a^- are away by some constant distance. We use the same loss function for the triplet (b, b^+, b^-). The sum of the two loss functions is defined as a bi-triplet loss.

The bi-triplet loss enforces features of a and a^+ are close and features of b and b^+ are close. As a (resp. a^+) and b (resp. b^+) are adjacent points on S (resp. S^+), we define a smoothness loss which enforces both features of a and b and features of a^+ and b^+ are as close as possible.

Local transformations are learned from the PatchNet to align two local patches P_a and P_{a+} and align two local patches P_b and P_{b+} respectively. The two edge vectors (a, b) and (a^+, b^+) should be aligned in $\mathbb{R}^3$ after applying the local transformations. Thus a transformation loss is defined on the two local transformations to enforce the spatial alignment of the two edge vectors.

We define the metric loss as the sum of the above loss functions, which is used for training the whole network. The metric loss considers the edge-wise correspondences in both feature and Euclidean spaces. Thus, the learned features encode local context and global structure.

Learning. A large corpus of bi-triplet pairs is constructed to train the EdgeNet and learn the network parameters. Specifically, we sample pairs of shapes. For each pair of shapes, we feed all pairs of corresponding edges into the network.

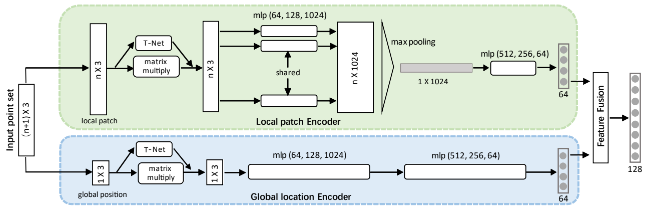


Figure 3: PatchNet architecture. Our model processes the local patch and global position through a deep multi-layer preceptron to extract local context- and global position-aware feature. mlp is the shorthand of multilayer perceptron (layer sizes are shown in bracket). All layers have Batchnorm and ReLU operations.

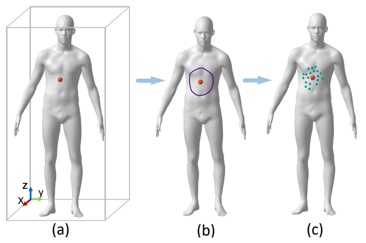


Figure 4: Construction of the local patch at the red vertex. (a) The coordinates of the vertex according to the bounding box of the shape is used as its global information; (b) The biharmonic distance circle (in purple) around the vertex; (c) N points are uniformly sampled in the biharmonic distance circle to form the local patch at the vertex.

3 RESULTS AND APPLICATIONS

We evaluate the discriminative power of our local shape features by conducting several qualitative and quantitative experiments. Throughout these experiments, EdgeNet is trained using different datasets for different tasks, but always with the same default network setting as described in Section 2. These experiments are conducted on a computer with a 4.20GHz Intel(R) Core(TM) I7-7700K CPU and a Nvidia 1080Ti 11GB GPU.

3.1 Performance and Evaluation

Two quantitative criteria and one qualitative popular criterion are used to measure the performance of the local shape features: *Cumulative Match Characteristic* (CMC) [9], *Princeton Protocol Counting* (PRP) [7] and the feature map.

Quantitative evaluation. We conduct comparisons with several existing techniques, includig LMVCNN [5], JLCNN [3], HKS [14], PCA [6], and SDF [13]. Our method is denoted as "EdgeNet". We use the same training and testing datasets for EdgeNet, LMVCNN, and JLCNN. Note that all hyperparameters of the LMVCNN and JLCNN are the same as Huang et al. (2017) and Chen et al. (2018) respectively. Due to the fact that HKS, PCA, and SDF features are class-insensitive and do not need training processes, they are directly evaluated on the same testing dataset.

Figure 5 exhibits the performance of different features using CMC on the testing dataset. And the corresponding accuracy (PRP) of different features are shown in Figure 6. We can see that our learned feature significantly outperforms other features, including the learned descriptors LMVCNN and JLCNN, and the hand-crafted local features successfully used in 3D shapes processing. We also plot situations that are generated by using our EdgeNet on different testing shapes, as shown in Figure 7. Corresponding points have the same RGB color.

Qualitative evaluation. We also qualitatively measure the

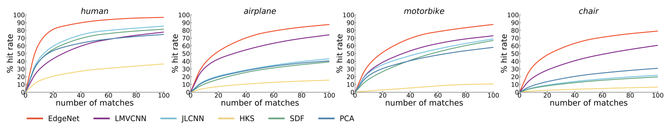


Figure 5: Performance of nearest neighbor matching in the deep feature space (CMC) measured using different features on each category. EdgeNet (red line) significantly outperforms the other standard features.

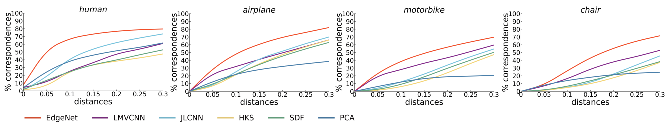


Figure 6: Performance of shape correspondence measured by using the PRP.

performance of different features using the feature map. We use t-SNE [10] to embed the local shape feature into a 2D space (perplexity = 30 by default). Figure 8 shows a two-dimensional visualization of different features computed as five points (head, left shoulder, right shoulder, knee, and thigh) across different shape transformations. Points with the same colors are semantical corresponding ones. The ideal situation is that the points of the same color should cluster as tight as possible. On the whole, our method can successfully discriminate the symmetrical points (sky blue sphere marked on left shoulder and dark blue sphere marked on right shoulder) across different shape transformations.

Ablation studies. To justify the specific design of PatchNet, we compare our full PatchNet against the network that omits the global location encoder. For each network, the performance of the learned feature are evaluated by computing the correspondence accuracy (PRP). Figure 9 (left) plots the results on the four categories. From the results, it can be observed that explicitly incorporating local and global information yields a better performance.

We also illustrate the necessity of transformation loss and smoothness loss in metric loss. The results of the correspondence accuracy (PRP) on airplane category demonstrate that our full metric loss function achieves the best performance, as shown in Figure 9 (right). This directly verifies the usefulness of smoothness loss and transformation term. Besides, the bi-triplet term, which is used to measure the semantic similarity in the feature space, also heavily contributes to the feature performance.

3.2 Applications

In this subsection, we apply our shape features in a wide range of geometric processing tasks, and evaluate it to the existing approaches. We conduct tasks of our shape features to model segmentation, partial-to-full matching, and shape retrieval.

Shape segmentation. Our network learns local shape feature

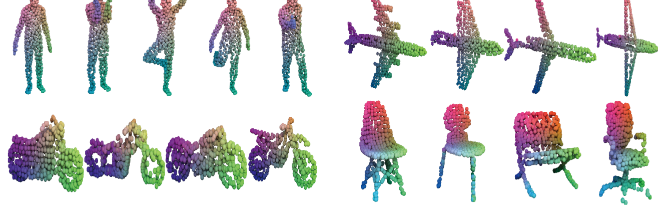
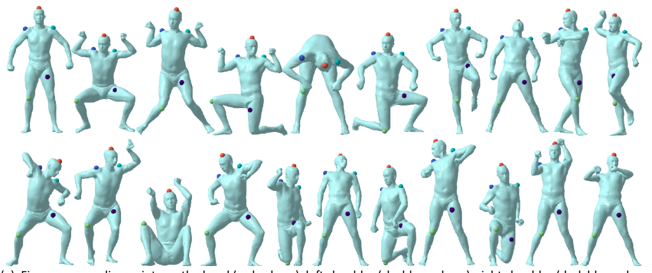
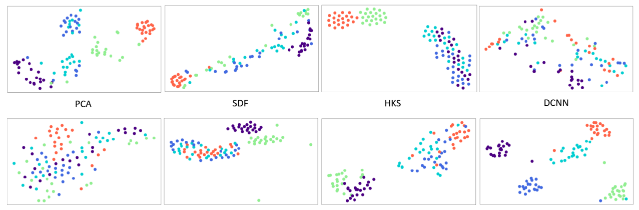


Figure 7: Visualization of point correspondences obtained with the learned local feature on different testing shapes. Corresponding points have the same RGB color.



(a) Five corresponding points on the head (red sphere), left shoulder (sky blue sphere), right shoulder (dark blue sphere), knee (green sphere), and thigh (purplesphere) across different shapes (21 examples from SCAPE testing dataset).



(b) 2D visualization of different feature spaces using t-SNE on human testing set. Each dot corresponds to a feature at a specific point on the shape marked in the respective color. The points with the same color should ideally cluster as tight and as possible).

Figure 8: Local feature visualization on SCAPE dataset.

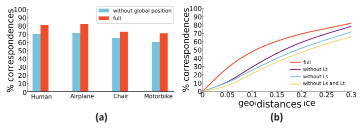


Figure 9: Ablation studies on different terms in (a) PatchNet and (b) metric loss.

that can be used for shape segmentation. The task is to assign category labels to different parts of an input shape by using our features. It is posed as a point-wise classification problem. We thus use a simple 1-layer multilayer perceptron on the part classification task. The features generated by pre-trained EdgeNet are used as input. We also make comparisons with PointNet [11], PointCNN [12], Guo et al. [4], and Kalogerakis et al. [6].

In Table 1, we report the labeling accuracy on the test shapes for all of the methods. The labeling accuracy improves 6.68% on average with our learned local shape feature. We also present some examples of shape segmentation results, as shown in Figure 10.

Partial matching. In the partial matching application, we are interested in generating dense matchings between partial shapes and full shapes. For each category, we train EdgeNet on the complete shapes with the same training dataset and configurations as above experiments, and then we extract the local features of testing partial models. As shown in Figure 11, we visualize the dense matches between partial-to-full shapes. Each point is rendered as a small ball.

Shape retrieval. Inspired by PointNet [11], we can aggregate all per-point local features of a shape, generated by our method, via a symmetric function *max pooling* into a 128-dim vector as its global feature, which can be used for shape re-

	mean	aero	chair	motor
ToG[2010]	73.83%	73.73%	71.94%	75.83%
ToG[2015]	67.97%	69.81%	59.15%	74.96%
PointNet	74.26%	70.14%	78.09%	74.54%
PointCNN	70.93%	70.81%	70.48%	71.52%
EdgeNet	**80.94%**	**77.45%**	**83.33%**	**82.03%**

Table 1: Shape labeling accuracy of different methods. ToG[2010] is short for Kalogerakis et al.[2010], and ToG[2015] is short for Guo et al.[2015].

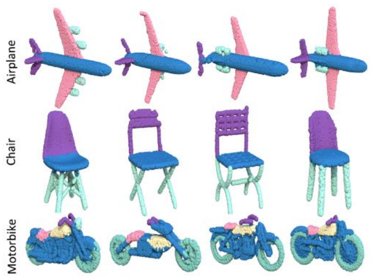


Figure 10: Visualizations of shape segmentation based on our learned feature. For each category, the corresponding parts are shown in same color.

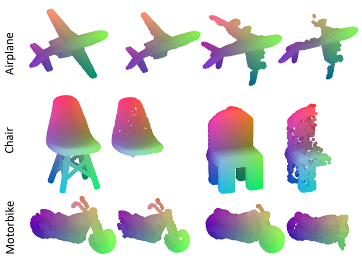


Figure 11: Dense correspondences of partial shapes with 3D complete shapes. Corresponding points have the same RGB color.

trieval. We compare our global feature with previous methods including PointNet [11] and PointCNN [8]. Figure 12 visualizes 2D embedding of learnt global shape features by using t-SNE (perplexity = 30 by default). It shows that our global shape feature appears visually more plausible than other methods.

4 CONCLUSIONS

In this article, we propose a metric learning architecture for generating semantic features of 3D shapes. The core of our approach is EdgeNet. It is composed of two PatchNets, which are end-to-end feature extractors. We first feed the PatchNet with a number of local patch to extractor the shape local representation. Then, we use the EdgeNet for feature and metric learning. In the process of local feature learning, the correct correspondence is kept between an edge-pair in feature space, by using the EdgeNet will in turn make the learned feature is structure-aware. And the PatchNet also encodes the local context and global position into the learned feature. Thus, we refer to our local descriptor as context-, asymmetry-, and structure-aware feature.

It is worth noting that EdgeNet is a *general-purpose* metric learning architecture, which can learn useful representation explicitly with well-defined relative relationships. We believe there will be much potential by applying EdgeNet in other practical applications.

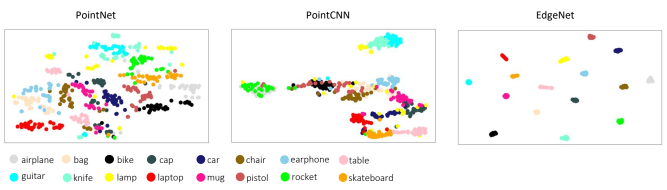


Figure 12: 2D visualization of different feature space by using t-SNE techniques. Each color corresponds to a shape category. Idea feature space would generate an 2D embedding map with dots of the same color cluster as tight as possible.

References

[1] Federica Bogo, Javier Romero, Matthew Loper, and Michael J Black. Faust: Dataset and evaluation for 3d mesh registration. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3794–3801, 2014.

[2] Alexander M Bronstein, Michael M Bronstein, Leonidas J Guibas, and Maks Ovsjanikov. Shape google: Geometric words and expressions for invariant shape retrieval. *ACM Transactions on Graphics*, 30(1):1, 2011.

[3] Mingjia Chen, Changbo Wang, and Qin Hong. Jointly learning shape descriptors and their correspondence via deep triplet cnns. *Computer Aided Geometric Design*, 62:192–205, 2018.

[4] Kan Guo, Dongqing Zou, and Xiaowu Chen. 3D mesh labeling via deep convolutional neural networks. *ACM Transactions on Graphics*, 35(1):3, 2015.

[5] Haibin Huang, Evangelos Kalogerakis, Siddhartha Chaudhuri, Duygu Ceylan, Vladimir G. Kim, and Ersin Yumer. Learning local shape descriptors from part correspondences with multiview convolutional networks. *ACM Transactions on Graphics*, 37(1), 2017.

[6] Evangelos Kalogerakis, Aaron Hertzmann, and Karan Singh. Learning 3d mesh segmentation and labeling. In *ACM Transactions on Graphics*, page 102, 2010.

[7] Vladimir G Kim, Wilmot Li, Niloy J Mitra, Siddhartha Chaudhuri, Stephen DiVerdi, and Thomas Funkhouser. Learning part-based templates from large collections of 3d shapes. *ACM Transactions on Graphics*, 32(4):70, 2013.

[8] Yangyan Li, Rui Bu, Mingchao Sun, Wei Wu, Xinhan Di, and Baoquan Chen. Pointcnn: Convolution on x-transformed points. In *Advances in Neural Information Processing Systems*, pages 826–836, 2018.

[9] Roee Litman and Alexander M Bronstein. Learning spectral descriptors for deformable shape correspondence. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 36(1):171–180, 2014.

[10] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9(12):2579–2605, 2008.

[11] Charles. R Qi, Hao Su, Kaichun Mo, and Leonidas J. Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 77–85, 2017.

[12] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In *Advances in Neural Information Processing Systems*, pages 5105–5114, 2017.

[13] Lior Shapira, Shy Shalom, Ariel Shamir, Daniel Cohen-Or, and Hao Zhang. Contextual part analogies in 3d objects. *International Journal of Computer Vision*, pages 309–326, 2010.

[14] Jian Sun, Maks Ovsjanikov, and Leonidas Guibas. A concise and provably informative multi-scale signature based on heat diffusion. In *Computer Graphics Forum*, volume 28, pages 1383–1392, 2009.

[15] Xiaolong Wang and Abhinav Gupta. Unsupervised learning of visual representations using videos. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2794–2802, 2015.

Mesh Learning Using Persistent Homology on the Laplacian Eigenfunctions

Yunhao Zhang [1], Haowen Liu [1], Paul Rosen [2], and Mustafa Hajij [1]

[1] *Ohio State University*
[2] *University of South Florida*

Abstract

We use persistent homology along with the eigenfunctions of the Laplacian to study similarity amongst triangulated 2-manifolds. Our method relies on studying the lower-star filtration induced by the eigenfunctions of the Laplacian. This gives us a shape descriptor that inherits the rich information encoded in the eigenfunctions of the Laplacian. Moreover, the similarity between these descriptors can be easily computed using tools that are readily available in Topological Data Analysis. We provide experiments to illustrate the effectiveness of the proposed method.

1 Introduction

Shape similarly is a critical problem is computer vision, geometric data processing and computer graphics. Multiple attempts have been made to quantify the similarity among 3D shapes [1, 30, 27]. Several challenges rise up when trying to construct an effective and efficient similarity measure including the complexity of the data, the potential noise in the data and the variation in the structure.

While the Laplacian eigenfunctions have been utilized in the literature of geometric processing to extract shape descriptors [29, 33], most of the eigenfunction-based descriptors require extensive processing to obtain an effective descriptor. Furthermore, the comparison between such descriptors requires designing a specialized similarity measure that adds to overhead computational time [32].

The eigenfunctions of the Laplacian store important information about the geometry of the underlying manifold [23, 24]. Moreover, spaces that have similar structures also tend to have similar sets of eigenfunctions [17]. From this perspective it is natural to utilize the eigenfunctions to measure the similarity between a collection of 3D shapes. The difficulty usually lies in finding the correspondence between two given manifolds [32]. More specifically, when manifold is discretized this correspondence might not even exist due to the difference between the cardinalities of the two vertex sets. Instead sub part correspondence might be considered, which is also a difficult problem [2].

In recent years the interplay between machine learning and Topological Data Analysis (TDA) has witnessed many developments with the better understanding of two tools in TDA: *Persistent Homology* (PH) [8] and the construction of *Mapper* [26]. These TDA tools have been shown to be a powerful tool for shape classification and recognition [15, 21], data summary [12, 5], topological signatures of data [3], graph understanding [13], among others.

In this paper we utilize persistent homology to extract the information encoded in the eigenfunctions of the Laplacian to obtain a topological mesh signature that can be used to measure the similarity among triangulated manifolds. Our proposed method has multiple advantages. On one hand, the method proposed here avoids the correspondence problem all together. Our approach relies on extracting the topological information encoded into the *lower-star filtration* (see Section 2 for the definition) of the eigenfunctions of the Laplacian and storing the resulting finger print in a structure called the *persistence diagram* [8]. This ultimately allows for an effective comparison between two manifolds by comparing between the persistence diagrams that are induced by the eigenfunctions of the Laplacian.

Using the persistence diagram to compare between metric spaces has been previously applied to meshes [4]. However, the metric-based method in [4] has two major limitations. Firstly, finding the distance function on large meshes is computationally expensive and usually requires utilizing a sampling technique, which might affect the quality of the final persistence diagram. Secondly, in order to obtain a strong descriptor from the persistence diagram induced by the distance matrix, one usually needs the information encoded in higher order persistence diagrams, which are expensive to compute.

Our method avoids these two limitations. On one hand, our method computes the persistence diagram using the lower-star filtration of one or a few eigenfunctions of the Laplacian. In fact we show that utilizing a single eigenfunction yields a persistence diagram that has more classification power than the metric-based approach in [4]. On the other hand, our method only requires the 0-order persistence diagram, which is very efficient to compute. We demonstrate our results by showing the effectiveness of our descriptor on standard datasets. See Section 3 for more details.

2 Persistence Homology on Triangulated Meshes

The mesh topological signature that we propose here utilizes a particular filtration that is induced by a scalar function defined on a mesh M. Our work is mainly aimed at studying triangulated meshes. However, we will state our definitions in terms of simplicial complexes. The reason for this is that most techniques introduced in this article are applicable beyond meshes, and we will provide more details in this regard towards the conclusion.

Let K be a simplicial complex. Let S be an ordered sequence $\sigma_1, \cdots, \sigma_n$ of all simplices in the complex K, such that for simplex $\sigma \in K$ every face of σ appears before in S. Then S induces a nested sequence of subcomplexes called a *filtration*:

$$\phi = K_0 \subset K_1 \subset ... \subset K_n = K \tag{2.1}$$

such that $K_i = \cup_{j \leq i} \sigma_j$ is the subcomplex obtained from first i simplicies $\sigma_1, \cdots, \sigma_i$ of S. Given a filtration as in 2.1, one may apply the homology functor on it to obtain a sequence of homology groups connected by homomorphism maps induced by the inclusions:

$$\mathcal{F}(K) : H_d(K_0) \longrightarrow H_d(K_1) \longrightarrow ... \longrightarrow H_d(K_n) \tag{2.2}$$

A d-homology class $\alpha \in H_d(K_i)$ is said to be *born* at time i if it appears for the first time as a homology class in $H_d(K_i)$. A d-class α *dies* at time j if it is trivial in $H_d(K_j)$ but not trivial in $H_d(K_{j-1})$. The *persistence* of the class α that is born at $H_d(K_i)$ and dies at $H_d(K_j)$ is defined to be $j - i$. *Persistent homology* captures the birth and death events in a given filtration and summarizes them in a multi-set structure called the *persistence diagram* $P^d(K)$ [8]. Specifically, the d-persistence diagram of a filtration $\mathcal{F}(K)$ is a collection of pairs (i, j) in the plane, where each (i, j) indicates a d-homology class that is created at time i in the filtration $\mathcal{F}(K)$ and dies entering time j. A persistence diagram can be represented equivalently by *persistence barcodes* [11]. Specifically every point (i, j) in the persistence diagram can be represented by a *bar* that starts at time u and ends at time v.

Persistence homology tracks the evolution of homology classes as this element moves though the homomorphism from left to right. More specifically, Persistent homology can be defined given any filtration, such as equation 2.1. For our purpose we are given a piece-wise linear function $f : |K| \longrightarrow \mathbb{R}$ defined on the vertices of K. We assume that the function f has different values on different vertices of K. Any such a function induces a filtration called the *lower-star* filtration. We define this filtration next.

Let $v \in V(K)$ be a vertex of K. The *star* of v, denoted as $St(v)$, is the set of all simplices in K that contain v as a vertex. When we are given a piece-wise linear function f defined on K, we can also define the lower star of v. Namely, the lower star of a vertex $v \in V(K)$ as $LowSt(v) = \{w \in St(v) | f(w) \leq f(v)\}$.

Let $V = \{v_1, \cdots, v_n\}$ be the set of vertices of K sorted in non-decreasing order of their f-values. Let $K_i := \cup_{j \leq i} LowSt(v_j)$. The lower-star filtration is the filtration is defined to be

$$\mathcal{F}_f(K) : \phi = K_0 \subset K_1 \subset ... \subset K_n = K \tag{2.3}$$

The lower-star filtration reflects the topology of the function f in the sense that the persistence homology induced by the filtration, equation 2.3, is identical to the persistent homology of the sublevel sets of the function f. We denote $P^d_f(K)$ to be the d-persistence diagram induced by the lower-star filtration $\mathcal{F}_f(K)$. In our work, the lower-star filtration is the main tool to extract the signature from a given space.

Here we focus on triangulated meshes, and we only compute the 0-persistence diagram on those meshes using the filtration induced by the lower-star filtration of the eigenfunctions of the Laplacian of these meshes. Such persistence diagrams can be efficiently computed using the *union-find* data structure.

2.1 The Lower-Star Filtration Induced by the Eigenfunctions of the Laplacian.

Let M be an triangulated manifold. The matrix L is self-adjoint and positive semi-definite. It has an *orthonormal eigensystem* $(\lambda_n, \phi_n)^{+\infty}_{n=0}$, $L\phi_n = \lambda_n \phi_n$, with $0 = \lambda_0 \leq \lambda_n \leq \lambda_{n+1}$, in $C(G)$. The eigenvectors of the Laplacian L form a rich family of scalar functions defined on G that have been utilized extensively in shape understanding and shape comparison [16, 23, 18]. The eigenfunctions of the Laplacian has also been used in graph understand [25], segmentation [22], and spectral clustering [20].

The reasons for extracting the information of the eigenfunctions of the Laplacian using the lower-star filtration can be summarized in the following points:

- The eigenfunctions of the Laplacian provide canonical scalar functions that depend only on the intrinsic geometric properties of the mesh. In other words, they have all desirable properties of eigenfunctions of the Laplacian—being invariant under certain deformation and robustness to noise and structure variation—will be inherited by the persistence diagram induced by the lower-star filtration of these functions.
- The eigenfunctions of the Laplacian store rich information about the geometry of the underlying manifold and the lower-star filtration provide the means to extract this information and stores it in the structure of the persistence diagram. This structure provides a ranking for features extracted from the eigenfunctions via the notion of persistence.

Ordering the eigenvectors of L by the increasing value of their corresponding eigenvalues, we use the first k-eigenvectors that correspond to the smallest nonzero k eigenvalues of L. These vectors contain low frequency information about the underlying manifold, and they usually retain the shape of complex meshes. In particular, we found that the first non-trivial eigenfunction of the Laplacian to be very effective for our purpose. This eigenfunction, called the *Fiedler vector* [9, 10], has many applications in graph theory as well as in computer graphics [14, 19]. Moreover, this vector has multiple interesting features. For instance, the maximum and the minimum of the Fielder vector tend to occur at points in the dataset with maximum geodesic distance [6] allowing its values to spread from one end of the graph following its "shape" to the other end.

2.2 Comparing Between Two Persistence Diagrams

We can quantify the structural differences persistence diagrams by using the bottleneck distance.

Let η be a bijection between two persistence diagrams X and Y. The *bottleneck distance* between X and Y [7] is defined as

$$W_\infty(X, Y) = \inf_{\eta : X \to Y} \sup_{x \in X} \|x - \eta(x)\|_\infty . \tag{2.4}$$

The bottleneck distance requires the persistence diagrams to have the same cardinalities. For this reason we allow infinitely replication of points along the diagonal $y = x$ to a given persistence diagram.

Other distances can also be defined on the space of persistence diagrams, such as the Wasserstein distance. For the purpose of this article we only restrict ourselves with the bottleneck distance.

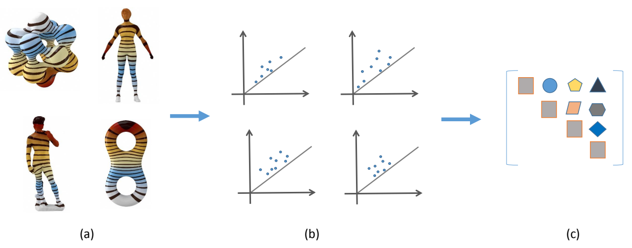


Figure 1: An illustration of the pipeline. (a) We compute one of the eigenfunctions of the Laplacian on the meshes that we want to compare. (b) The lower-star filtrations of the meshes with respect to these scalar functions are computed and their persistence diagrams are extracted. (c) A pairwise comparison between the persistence diagrams is performed using the bottleneck or Wasserstein distances.

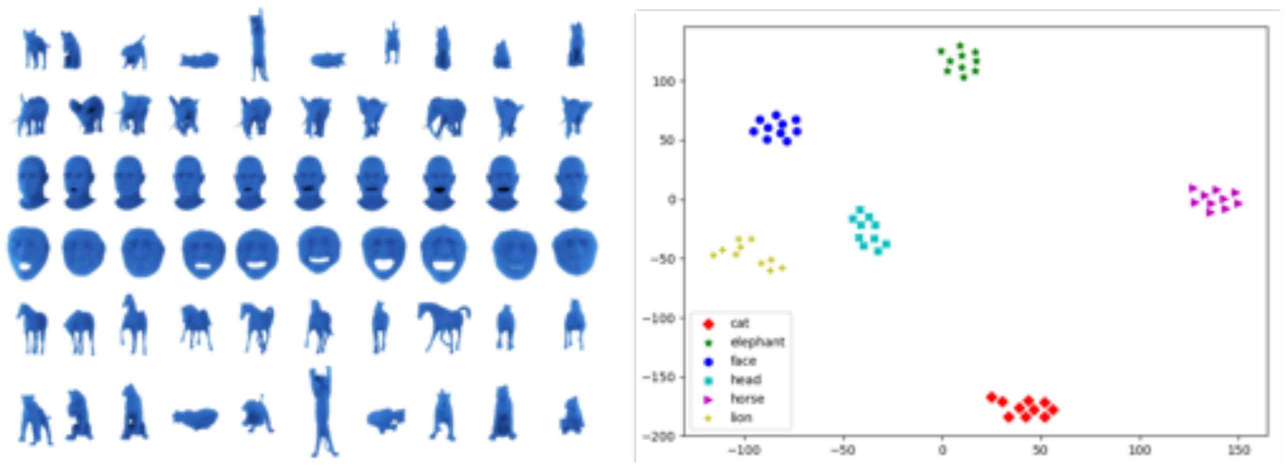


Figure 2: On the left we show the data set that we used in our experiments. The data set consists of 60 triangulated meshes that are divided into 6 categories, which are shown in the figure on the right. We compute the Fielder's vector for each mesh in this data set and then compute the 0-persistence diagram associated with the lower-star filtration of this vector. The bottleneck distance between these diagrams is calculated, and the figure on the right shows the 2d t-SNE projection obtained using the final distance matrix. Notice how our method provides distinct clusters on this data.

3 Method and Results

Given the above setup, our method can be summarized as follows. First we compute a certain eigenfunction of the Laplacian on a given mesh dataset. In our case we used the Fielder's vector. We then compute the 0-persistence diagrams of the lower-star filtration induced by the chosen eigenfunction. Once we have the persistence diagrams of the meshes, the distances between these diagrams can be computed using the bottleneck distance we defined in Section 2.2. See Figure 1 for a summary of the method.

To validate the effectiveness of the topological descriptor proposed here, we test it using a publicly available data from [28]. The data set consists of 60 meshes that are divided into 6 categories: cat, elephant, face, head, horse and lion. Each category contains exactly ten triangulated meshes.

On this dataset, we computing the distance matrix between the persistence diagram of the lower-star filtration of the induced Fielder's vectors using bottleneck distance. To assess the final results, we compute the 2d t-SNE projection [31] of final distance matrix. The result is reported in Figure 2.

Note how this topological descriptor provides a distinct clusters for the underlying data set. Furthermore, the results shown here shows that the proposed descriptor has a better classification power than the one proposed in [4].

One observation worth mentioning here is that the t-SNE projection in Figure 2 shows that heads and faces clusters appear to be closer to the lions cluster than cats cluster. The reason for this is mostly an artifact of the t-SNE projection. In fact the MDS projection shows that the lions and the cats cluster are indeed closer to each other than heads and faces. We presented the t-SNE projection here over MDS since the latter showed the some clusters too close to each other.

4 Further Directions and Conclusion

The experimentation results are only shown with respect to Fielder's vector. In theory any eigenfunction of the Laplacian can be used in a similar manner, as illustrated above. Combining the signatures obtained from multiple eigenfunction potentially provides even a stronger descriptor. We plan to pursue this direction in the extension of this work.

The construction that we introduced here on triangulated meshes can be easily extended to study similarity between other types of objects. Namely any domain where the definition of the Laplacian is applicable, such as points clouds and graphs. We plan to investigate these directions in the future.

References

[1] Silvia Biasotti, Andrea Cerri, Alexander M Bronstein, and Michael M Bronstein. Quantifying 3d shape similarity using maps: Recent trends, applications and perspectives. In *Eurographics (State of the Art Reports)*, pages 135–159, 2014.

[2] Silvia Biasotti, Simone Marini, Michela Spagnuolo, and Bianca Falcidieno. Sub-part correspondence by structural descriptors of 3d shapes. *Computer-Aided Design*, 38(9):1002–1019, 2006.

[3] Thomas Bonis, Maks Ovsjanikov, Steve Oudot, and Frédéric Chazal. Persistence-based pooling for shape pose recognition. In *International Workshop on Computational Topology in Image Context*, pages 19–29. Springer, 2016.

[4] Frédéric Chazal, David Cohen-Steiner, Leonidas J Guibas, Facundo Mémoli, and Steve Y Oudot. Gromov-hausdorff stable signatures for shapes using persistence. *Computer Graphics Forum*, 28(5):1393–1403, 2009.

[5] ANM Imroz Choudhury, Bei Wang, Paul Rosen, and Valerio Pascucci. Topological analysis and visualization of cyclical behavior in memory reference traces. In *2012 IEEE Pacific Visualization Symposium*, pages 9–16. IEEE, 2012.

[6] Moo K Chung, Seongho Seo, Nagesh Adluru, and Houri K Vorperian. Hot spots conjecture and its application to modeling tubular structures. In *International Workshop on Machine Learning in Medical Imaging*, pages 225–232, 2011.

[7] Herbert Edelsbrunner and John Harer. *Computational Topology: An Introduction*. American Mathematical Society, Providence, RI, USA, 2010.

[8] Herbert Edelsbrunner, David Letscher, and Afra J. Zomorodian. Topological persistence and simplification. *Discrete and Computational Geometry*, 28:511–533, 2002.

[9] Miroslav Fiedler. Algebraic connectivity of graphs. *Czechoslovak mathematical journal*, 23(2):298–305, 1973.

[10] Miroslav Fiedler. A property of eigenvectors of nonnegative symmetric matrices and its application to graph theory. *Czechoslovak Mathematical Journal*, 25(4):619–633, 1975.

[11] Robert Ghrist. Barcodes: The persistent topology of data. *Bullentin of the American Mathematical Society*, 45:61–75, 2008.

[12] Mustafa Hajij, Nataša Jonoska, Denys Kukushkin, and Masahico Saito. Graph based analysis for gene segment organization in a scrambled genome. *arXiv preprint arXiv:1801.05922*, 2018.

[13] Mustafa Hajij, Bei Wang, Carlos Scheidegger, and Paul Rosen. Visual detection of structural changes in time-varying graphs using persistent homology. In *2018 IEEE Pacific Visualization Symposium (PacificVis)*, pages 125–134. IEEE, 2018.

[14] Martin Isenburg and Peter Lindstrom. Streaming meshes. In *Visualization, 2005. VIS 05. IEEE*, pages 231–238. IEEE, 2005.

[15] Genki Kusano, Yasuaki Hiraoka, and Kenji Fukumizu. Persistence weighted gaussian kernel for topological data analysis. In *International Conference on Machine Learning*, pages 2004–2013, 2016.

[16] Stephane Lafon and Ann B Lee. Diffusion maps and coarse-graining: A unified framework for dimensionality reduction, graph partitioning, and data set parameterization. *IEEE transactions on pattern analysis and machine intelligence*, 28(9):1393–1403, 2006.

[17] Bruno Levy. Laplace-beltrami eigenfunctions towards an algorithm that" understands" geometry. In *IEEE International Conference on Shape Modeling and Applications 2006 (SMI'06)*, pages 13–13. IEEE, 2006.

[18] Bruno Lévy. Laplace-beltrami eigenfunctions towards an algorithm that understands geometry. In *IEEE International Conference on Shape Modeling and Applications*, page 13, 2006.

[19] Patrick Mullen, Yiying Tong, Pierre Alliez, and Mathieu Desbrun. Spectral conformal parameterization. *Computer Graphics Forum*, 27(5):1487–1494, 2008.

[20] Andrew Y Ng, Michael I Jordan, and Yair Weiss. On spectral clustering: Analysis and an algorithm. In *Advances in neural information processing systems*, pages 849–856, 2002.

[21] Jan Reininghaus, Stefan Huber, Ulrich Bauer, and Roland Kwitt. A stable multi-scale kernel for topological machine learning. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4741–4748, 2015.

[22] Martin Reuter, Silvia Biasotti, Daniela Giorgi, Giuseppe Patané, and Michela Spagnuolo. Discrete laplace–beltrami operators for shape analysis and segmentation. *Computers & Graphics*, 33(3):381–390, 2009.

[23] Martin Reuter, Franz-Erich Wolter, and Niklas Peinecke. Laplace–beltrami spectra as shape-dnaof surfaces and solids. *Computer-Aided Design*, 38(4):342–366, 2006.

[24] Raif M Rustamov. Laplace-beltrami eigenfunctions for deformation invariant shape representation. In *Proceedings of the fifth Eurographics symposium on Geometry processing*, pages 225–233. Eurographics Association, 2007.

[25] David I Shuman, Sunil K Narang, Pascal Frossard, Antonio Ortega, and Pierre Vandergheynst. The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains. *IEEE Signal Processing Magazine*, 30(3):83–98, 2013.

[26] Gurjeet Singh, Facundo Mémoli, and Gunnar E Carlsson. Topological methods for the analysis of high dimensional data sets and 3d object recognition. In *SPBG*, pages 91–100, 2007.

[27] Tomáš Skopal and Benjamin Bustos. On nonmetric similarity search problems in complex domains. *ACM Computing Surveys (CSUR)*, 43(4):34, 2011.

[28] Robert W Sumner and Jovan Popović. Deformation transfer for triangle meshes. *ACM Transactions on graphics (TOG)*, 23(3):399–405, 2004.

[29] Jian Sun, Maks Ovsjanikov, and Leonidas Guibas. A concise and provably informative multi-scale signature based on heat diffusion. *Computer graphics forum*, 28(5):1383–1392, 2009.

[30] Gary KL Tam, Zhi-Quan Cheng, Yu-Kun Lai, Frank C Langbein, Yonghuai Liu, David Marshall, Ralph R Martin, Xian-Fang Sun, and Paul L Rosin. Registration of 3d point clouds and meshes: a survey from rigid to nonrigid. *IEEE transactions on visualization and computer graphics*, 19(7):1199–1217, 2013.

[31] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-SNE. *Journal of Machine Learning Research*, 9:2579–2605, 2008.

[32] Oliver Van Kaick, Hao Zhang, Ghassan Hamarneh, and Daniel Cohen-Or. A survey on shape correspondence. *Computer Graphics Forum*, 30(6):1681–1707, 2011.

[33] Lisha Zhang, Manuel João da Fonseca, Alfredo Ferreira, and Combinando Realidade Aumentada e Recuperação. Survey on 3d shape descriptors. *FundaÃgao para a Cincia ea Tecnologia, Lisboa, Portugal, Tech. Rep. Technical Report, DecorAR (FCT POSC/EIA/59938/2004)*, 3, 2007.

3D Femur Skeletonization using Maximum-Minimum Centre Approach

Saw Seow Hui[1], Ewe Hong Tat[1], Ji Wan Kim[2], and Byung Gook Lee[3]

[1] *Universiti Tunku Abdul Rahman, Malaysia*
[2] *Asan Medical Center, University of Ulsan, College of Medicine, South Korea*
[3] *Division of Computer Engineering, Dongseo Univ., South Korea*

Abstract

Femoral shaft fractures have been correlated with frequent morbidity and mortality. It is a major musculoskeletal disorder caused by tremendous force being applied to the femur. One of the most common surgical treatments for fixation is the intramedullary nailing, which utilizes a specially designed metal rod and screws to be implanted into medullary canal. However, severe bowing of the femur can result in mismatch between the intramedullary nail and the alignment of the femur. Such mismatch is a risk factor for anterior cortical perforation of the distal femur with subtrochanteric fractures, and leg length discrepancy with fractures of the femoral shaft. Therefore, the exact data of the femur geometry is mandatory to develop and apply intramedullary nail for bowed femur. This research is to develop an automatic approach with direct extraction of the skeleton from a 3D femur for each individual patient, in order to produce an accurate 3D preoperative simulation possible. The 3D femur is generated from a set of computed tomography images. The efficiency and the robustness of 3D skeletonization based on maximum-minimum centre approach will be discussed in this paper. The proposed approach can potentially be assisted for the implant measurements. Several examples are included to demonstrate that the proposed approach works well for several 3D femurs.

1 Introduction

Three-dimensional (3D) skeletonization provides an alternative to capture the inner structure of an overall complex 3D mesh by forming its own skeletons. These computed skeletons consist of significant geometric and topological information that are used extensively to produce segmentation for various analyses and visualization in medical imaging, robotics and video surveillance applications.

Although the development of 2D skeletonization is relatively well established, but the skeletons computation in 3D is yet a challenging task for both researchers and practitioners. And, it is definitely worth to be explored in the study of orthopaedics especially the human femur.

The femur is the longest, heaviest and strongest bone in our human body. Different kinds of trauma with a lot of forces can damage this bone, such as in some motor vehicle accidents or motorcycle crash. This can also happen in a lower-force accident, such as fall from slippery floor, ladder landing on foot among the older people due to their weaker bones or osteoporosis. Femoral fractures usually require, open reduction internal fixation (ORIF) with intramedullary nail or plate to repair and heal the broken bones [3].

As for the ORIF, it consists of two procedures performed by an orthopaedic surgeon under anaesthesia: open reduction and internal fixation [1]. ORIF involves reduction of the fracture and apply an internal fixation device such as an intramedullary nail (usually made of titanium) into the medullary canal in order to stabilize the fracture until bone union. This procedure is also known as intramedullary nailing and involves the use of other special types of implants including metal plates, screws, stainless steel pins and wires.

Intramedullary nailing is one of the most common surgical treatment for femoral fracture fixation. Hence, the preoperative planning template is an essential prerequisite to estimate the correct nail diameter and length for the success of orthopaedic procedures [5].

However, mismatch problem of current available nail with bowed femurs and, an accurate and automatic 3D preoperative simulation are therefore desirable [4].

2 Skeletonization using Max-Min Center

There are mainly three processes to compute as illustrated in Figure 1. It begins with the individual snapshots of the individual human femur into a set of cross-sectional images, which also known as Computed Tomography (CT) imaging. These images are used to produce three-dimensional (3D) samplings of anatomy elements for the human femur. With the use of reconstruction and parametrization from such datasets, the structured data (in the form of *.obj* format) can be obtained to form a 3D model.

Computed Tomography (CT) Images
3D Femur Construction
Skeletonization

Figure 1: The three main processes.

The final process is the skeletonization to obtain the compact representation of the femur. The earliest approach for the skeleton extraction is the medial axis transform (MAT) proposed by Blum [2, 7]. MAT is mainly composed of two properties: the medial axis (MA) and the radius function. Medial axis (MA) for a given object Ω is defined as the loci of all maximal inscribed disks that meets two or more boundary points without crossing any of the boundaries.

$$MAT(\Omega) = \{(p, r) \in R^n \times R | B_r(p) \text{ is maximal ball in } \Omega\}$$

$$MA(\Omega) = \{p \in R^n | r \geq 0 \text{ s.t. } (p, r) \in MAT(\Omega)\}$$

In order to obtain a detailed information of the object, each point on the medial axis is associated with the radius function which form the medial circle. The volume enclosed by the surface of the object is exactly the union of these circles.

To obtain the skeleton of the 3D femur, we try to find maximum radius of circle that can fit into the irregular inner polygon of the slice 3D femur data. We use the following steps to obtain a reliable skeleton.

1. Sliding process : the sliding window with appropriate height h is decided in the beginning. Then, the vertices in the sliding window are collected,

$$V_s = \{v : v_z \in (z, z+h)\}$$

   where $v = (v_x, v_y, v_z) \in R^3$, $s = (z, z+h)$.

2. Adjacent face [6] : obtain the faces adjacent to the points in V_s,

$$F_s = \lceil\lceil V_s \rceil\rceil = \{f_i\}_{i=0}^{m}.$$

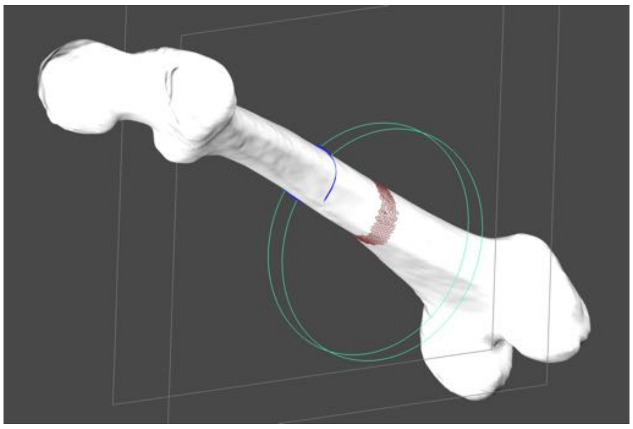


Figure 2: Sliding window (2 green circles), Adjacent face (brown triangle faces).

3. Weighted center point : calculate the weighted average of F_s,

$$\bar{F_s} = \frac{1}{A_s}\sum_{i=0}^{m} A(f_i)\bar{f_i}$$

   where $A_s = \sum_{i=0}^{m} A(f_i)$, $A(f_i)$ = area of a triangle f_i, and $\bar{f_i}$ = center of a triangle f_i.

4. Slice plane : calculate the weighted average of the face normal sliding window base plane : face normal average.

5. Inner slice polygon Ω_s : projection face center to base plane.

6. Max-Min Center : $B_s(p, r)$ is maximal circle in Ω_s.

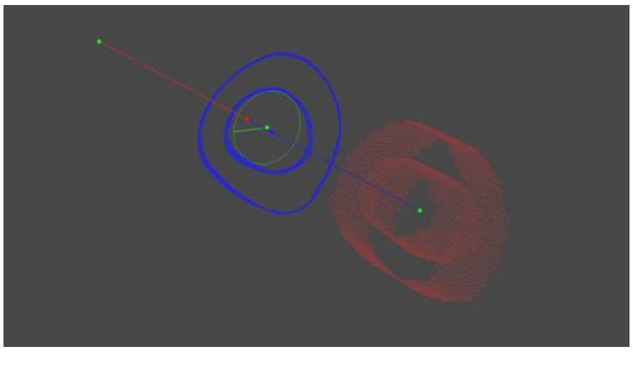


Figure 3: Weighted center point (green point), Inner slice polygon (blue points), Maximal circle $B_s(p, r)$ (green circle and line).

In this research, we have proposed maximum-minimum centre approach to obtain a reliable skeleton of the 3D femur. In the future, if a lot of relevant data is accumulated, we will develop a more efficient algorithm using artificial intelligence theory. The 3D femur skeletonization would give better understanding of the geometry and help to prepare and develop new intramedullary nailing system.

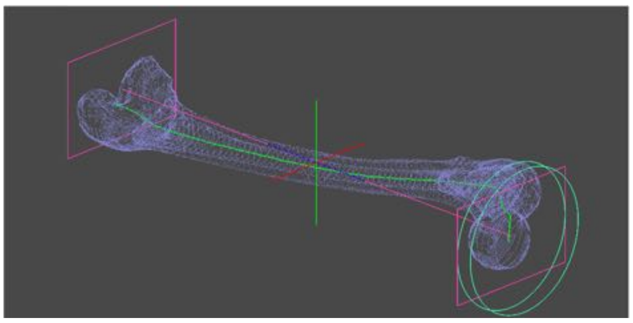


Figure 4: The 3D femur skeletonization result (green lines).

Acknowledgement

This work was supported by Institute for Information and Communications Technology Promotion(IITP) grant funded by the Korea government(MSIT) (No.2018-0-00245, Development of prevention technology against AI dysfunction induced by deception attack)

References

[1] Open reduction and internal fixation (orif). intermountainhealthcare.org, July 2012.

[2] Harry Blum et al. A transformation for extracting new descriptors of shape. *Models for the perception of speech and visual form*, 19(5):362–380, 1967.

[3] Yi-Bin Gao, Song-Lin Tong, Jian-Hao Yu, and Wen-Jie Lu. [case control study on open reduction internal fixation (orif) and minimally invasive percutaneous plate osteosynthesis (mippo) for the treatment of proximal humerus fractures in aged]. *Zhongguo gu shang = China journal of orthopaedics and traumatology*, 28(4):335—339, April 2015.

[4] Ji Wan Kim, Hyunuk Kim, Chang-Wug Oh, Joon-Woo Kim, Shon oog jin, Young-Soo Byun, Jung Jae Kim, Hyoung Keun Oh, Hiroaki Minehara, Kyu-Tae Hwang, and Ki Chul Park. Surgical outcomes of intramedullary nailing for diaphyseal atypical femur fractures: is it safe to modify a nail entry in bowed femur? *Archives of Orthopaedic and Trauma Surgery*, 137, 08 2017.

[5] Atesok Kivanc, Galos David, M. Jazrawi Laith, and A. Egol. Kenneth. Preoperative planning in orthopaedic surgery: current practice and evolving applications. Technical Report 4, 2015.

[6] Peter Lindstrom and Greg Turk. Evaluation of memoryless simplification. *IEEE Transactions on Visualization and Computer Graphics*, 5(2):98–115, 1999.

[7] Jaehwan Ma, Sang Won Bae, and Sunghee Choi. 3d medial axis point approximation using nearest neighbors and the normal field. *The Visual Computer*, 28(1):7–19, 2012.

Triangular Trigonometric Patches for Surface Interpolant

Xiang Fang [1] and Stephen Mann [1]

[1] *Cheriton School of Computer Science, University of Waterloo, Waterloo, ON N2L 3G1 CANADA*

Abstract

We construct a new triangular patch with ten control points weighted by trigonometric functions. This patch interpolates the positions and normals of the corners of a triangle, and has the same C^1 continuity conditions as a triangular cubic Bézier patch. Furthermore, we can construct a hybrid variant, giving four center control points: one copy of the center control point is used to achieve C^1 continuity across each boundary, while the fourth center control point is a shape parameter that does not affect the C^1 conditions across any of the boundaries.

1 Introduction

A cubic triangular Bézier patch is defined as (see Fig. 1)

$$b^n(\boldsymbol{u}) = \sum_{|\boldsymbol{i}|=n} p_{\boldsymbol{i}} B^n_{\boldsymbol{i}}(\boldsymbol{u}),$$

where $p_{\boldsymbol{i}}$ are control points, $B^n_{\boldsymbol{i}}(\boldsymbol{u})$ are the *bivariate Bernstein polynomials*,

$$B^n_{\boldsymbol{i}}(\boldsymbol{u}) = \binom{n}{\boldsymbol{i}} u^i v^j w^k,$$

with $\boldsymbol{u} = (u, v, w)$ being barycentric coordinates relative to a domain triangle, $\boldsymbol{i} = (i, j, k)$ is a multi-index with $n = |\boldsymbol{i}| = i + j + k$, where $\binom{n}{\boldsymbol{i}} = \frac{n!}{i!j!k!}$. In this paper, we use u and v as parameters with $w = 1 - u - v$. The barycentric coordinates $\boldsymbol{u}$ can be written as $\boldsymbol{u} = (u, v)$.

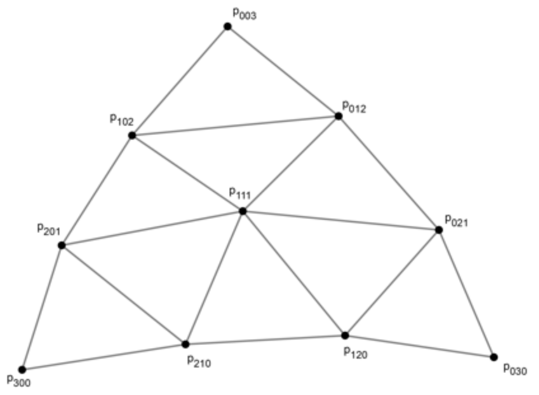


Figure 1: Control point layout of a cubic triangular Bézier/trigonometric patch.

Our cubic triangular trigonometric patch uses the same layout of control points as the Bézier patch; its blending functions are

$$\begin{aligned}
f_{300}(\boldsymbol{u}) &= (1-d)^2,\\
f_{210}(\boldsymbol{u}) &= 2b(1-d)f,\\
f_{201}(\boldsymbol{u}) &= 2c(1-d)e,\\
f_{120}(\boldsymbol{u}) &= 2a(1-e)f,\\
f_{111}(\boldsymbol{u}) &= 2abc,\\
f_{102}(\boldsymbol{u}) &= 2ae(1-f),\\
f_{030}(\boldsymbol{u}) &= (1-e)^2,\\
f_{021}(\boldsymbol{u}) &= 2cd(1-e),\\
f_{012}(\boldsymbol{u}) &= 2bd(1-f),\\
f_{003}(\boldsymbol{u}) &= (1-f)^2.
\end{aligned} \tag{1.1}$$

where

$$\begin{aligned}
a &= \sin\tfrac{\pi w}{2},\\
b &= \sin\tfrac{\pi u}{2},\\
c &= \sin\tfrac{\pi v}{2},\\
d &= \sin\tfrac{\pi(u+v)}{2},\\
e &= \sin\tfrac{\pi(v+w)}{2},\\
f &= \sin\tfrac{\pi(w+u)}{2}.
\end{aligned} \tag{1.2}$$

2 Properties of the Trigonometric Patch

The cubic triangular trigonometric patch is similar to the cubic triangular Bézier patch: the patch interpolates the locations and normals of the corners of a triangle, and the normals of the corners of the resulting surface are determined by the triangle panels (for example $\triangle p_{300}p_{210}p_{201}$ in Fig. 1).

2.1 Continuity

Two triangular trigonometric patches over neighbouring triangles meet with C^1 continuity if they meet with C^0 continuity (sharing a common boundary) and if adjacent panels are coplanar. The control points involved in the C^1 continuity conditions across a boundary are the same in our patch as they are for polynomial Bézier patch. Thus any scheme that constructs cubic triangular Bézier patches that meet with C^1 continuity could use our patch instead of cubic triangular Bézier patches while still using the same construction to determine the control points.

See [1] for additional details on triangular Bézier patches.

3 Blended Center Control Point

We divide the blending function of the center control point (p_{111} in Fig. 1) into four sub-functions. Assigning each of each sub-function to a new control point, the new blending functions of this new patch are

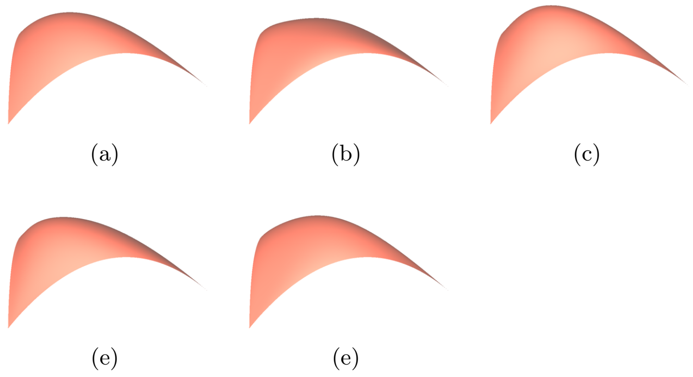


Figure 2: (a) The original patch surface. (b) Shift the center point downward. (c) Shift the center point upward. (d) Shift the center point left. (e) Shift the center point right.

$$
\begin{array}{lcl}
f_{300}(\boldsymbol{u}) & = & (1-d)^2, \\
f_{210}(\boldsymbol{u}) & = & 2b(1-d)f, \\
f_{201}(\boldsymbol{u}) & = & 2c(1-d)e, \\
f_{120}(\boldsymbol{u}) & = & 2a(1-e)f, \\
f_{111-100}(\boldsymbol{u}) & = & 2abc(\frac{b^2c^2}{e^2}+\frac{b^2c^2}{f^2}), \\
f_{111-010}(\boldsymbol{u}) & = & 2abc(\frac{a^2c^2}{d^2}+\frac{a^2c^2}{f^2}), \\
f_{111-001}(\boldsymbol{u}) & = & 2abc(\frac{a^2b^2}{d^2}+\frac{a^2b^2}{e^2}), \\
f_{111-000}(\boldsymbol{u}) & = & 2abc - f_{111100}(\boldsymbol{u}) - f_{111010}(\boldsymbol{u}) - f_{111001}(\boldsymbol{u}), \\
f_{102}(\boldsymbol{u}) & = & 2ae(1-f), \\
f_{030}(\boldsymbol{u}) & = & (1-e)^2, \\
f_{021}(\boldsymbol{u}) & = & 2cd(1-e), \\
f_{012}(\boldsymbol{u}) & = & 2bd(1-f), \\
f_{003}(\boldsymbol{u}) & = & (1-f)^2.
\end{array}
\tag{3.3}
$$

It is impossible to interpolate the positions and normals of triangular mesh over a plane with cubic triangular Bézier patches. However, with the blended center point triangular trigonometric patch, the three of the four blended center points are related to one boundary each. Each of these three center points only affects the C^1 continuity condition of one boundary and does not affect the continuity across the other two boundaries. The last blended center point is a "free" control point, and does not affect any continuity conditions across any boundary. Figure 2 shows five different patch surfaces that are variants of an original triangular trigonometric patch. All of them share the same boundaries, and the same first order partial derivative values along all boundaries.

Our resulting patch can meet its neighbours with C^1 continuity across all three boundaries. However, as is common with hybrid schemes, the blending functions have a singularity at the corners. This singularity is removable for position and first derivative (and thus, our patch interpolates the position and normal data of the input triangle), but the patch does not have well defined second derivatives at the corners.

4 Example

We devised a simple data fitting scheme to test the ideas in this abstract. The scheme starts by constructing cubic Bézier curves to interpolate the data at pairs of each input triangle, giving settings for control points $p_{300}, p_{210}, p_{120}, p_{030}, p_{021}, p_{012}, p_{003}, p_{102}, p_{201}$ for the triangular Bézier patch of Fig. 1. We then set p_{111} to an initial value of

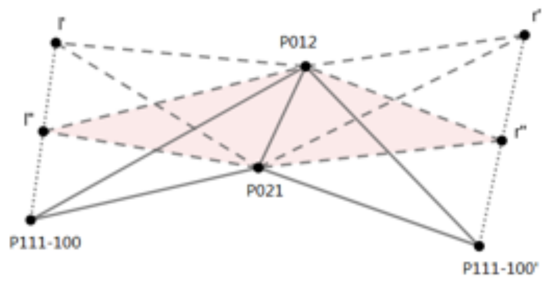


Figure 3: An example of blending the control points across a boundary.

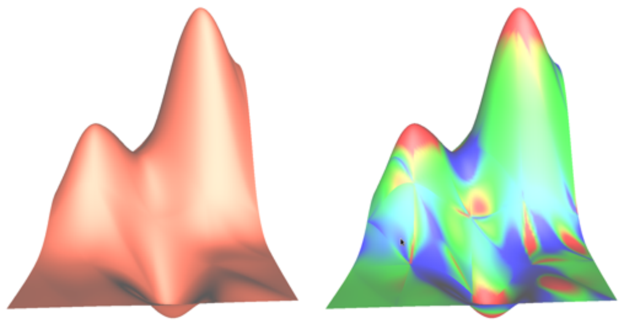


Figure 4: Trigonometric surface and its Gaussian curvature plot (curvature computed numerically).

$(p_{210}+p_{120}+p_{021}+p_{012}+p_{102}+p_{201})/6$. A network of patches constructed in this manner will meet with only C^0 continuity.

As a second step, we split each center control point p_{111} of each patch into four. Three of copies of this control point will be used to achieve C^1 continuity across the three boundaries of the patch. For example, to obtain C^1 continuity across the boundary between p_{030}, p_{003}, we extend the p_{111} control points on either side of this boundary to points $p_{111-100}$ and $p_{111-100'}$, and average them to find the location of one of the p_{111} center control points (l'' for this patch) for the patch (Figure 3). The fourth center control point is set to be the average of the other three copies of the center control points.

Then, there are additional steps applied to improve the quality of the surfaces. The basic idea is to make the four center control points get closer, thus the resulting surface will be closer to a Bézier surface. In particular, we apply the following two steps in turn for several times.

- Set the values of the first three center control points to be the value of the fourth center control point.
- Across each boundary, blend the values of two center control points (Figure 3) to meet C^1 continuity.

An example of a surface constructed with our method for a sampling of the Frankye function [2] is shown in Figure 4.

References

[1] G. Farin. *Curves and Surfaces for CAGD.* Morgan-Kaufmann, fifth edition, 2001.

[2] R. Franke. A critical comparison of some methods for interpolation of scattered data. Technical Report NPS-53-79-003, Naval Postgraduate School, 1979.

Quasi arc-length approximation with cubic B-splines

Javier Sánchez-Reyes [1], Jesús M. Chacón [1], and Rubén Dorado [2]

[1] *Universidad de Castilla-La Mancha (Spain)*
[2] *Universidad de Jaén (Spain)*

Abstract

We present a simple, yet general tool for curve approximation with quasi arc-length parameterization, having in mind the specification of trajectories for CNC machining or 3D-printing. We consider a low degree curve that admits exact implementation as G-code, namely a cubic B-spline with two internal knots that can be freely located. The quasi arc-length condition is incorporated in Hermite fashion, by imposing unit speed and vanishing tangential acceleration at the endpoints. This results in a straightforward geometric constraint on the lengths of the projections of the control legs. By additionally prescribing the position, tangent direction and curvature at the endpoints, this tool can be employed for osculatory Hermite interpolation.
We give examples of application for generating polynomial quasi-arc length approximations of conics, or transcendental curves such as the clothoid. Finally, we explore the potential of this scheme for free-form design, by moving the control points in a constrained manner and adjusting automatically the internal knots to meet the quasi-arc length condition.

1 Why quasi arc-length approximations

Curves in commercial CAD systems are usually represented by using a polynomial (or rational) parameterization. However, the standard rational model suffers from two shortcomings:

1. It does not encompass remarkable cases, such as the offset or transcendental curves, which, consequently, must be approximated in some way.

2. It cannot yield curves with exact arc-length parameterization [3], aside from a straight line. An approximation with improved parameterization may be required, even for curves in the model, such as a humble circle [1, 5].

Most approximation methods, and especially those based on geometric continuity techniques, tackle only the first point and concentrate on generating a good approximant in the sense of being close to a given curve. However, this requisite does not suffice for applications requiring a smooth parameterization, i.e., at least C^2 and close to the ideal arc-length (natural) parameterization. Parameterization plays a key role in a trajectory whose parameter is taken as proportional to time. A smooth parameterization with approximately unit speed is desirable for CNC machining and 3D-printing, to improve the surface quality, or trajectories for robotic manipulators. Parameterization is also of paramount importance in *skinning* [4] (also known as *lofting*), because unevenly parameterized sections result in surfaces that display poor quality.

Our approach addresses both shortcomings simultaneously, approximating those curves that do not fit into the NURBS model, with the bonus of a smooth parameterization.

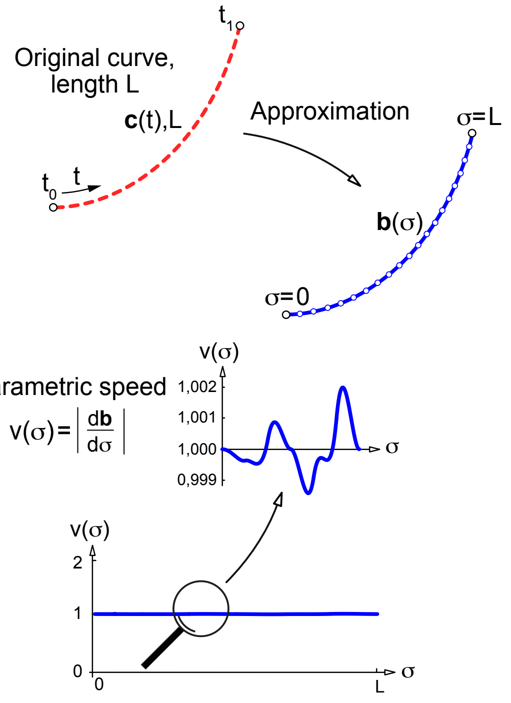


Figure 1: Quasi arc-length approximation $\mathbf{b}(\sigma)$ to a curve $\mathbf{c}(t)$.

Mathematically, the approximation we advocate is sketched in Fig. 1. Given an original curve segment $\mathbf{c}(t)$, with arbitrary parameterization over a general domain $t \in [t_0, t_1]$, we seek a quasi arc-length approximation $\mathbf{b}(\sigma)$, defined over an interval $\sigma \in [0, L]$ of length L equal to that of $\mathbf{c}(t)$, where σ approximates the arc-length parameter s. Therefore, the normalized parametric speed $v(\sigma)$ should be close to the ideal unit speed:

$$v(\sigma) = \left|\frac{d\mathbf{b}}{d\sigma}\right| \approx 1, \quad \sigma \in [0, L]. \tag{1}$$

2 Piecewise Hermite interpolation

To achieve a quasi arc-length approximation to $\mathbf{c}(t)$, a first option would be to sample points on it and construct an interpolant that tries to minimize its speed deviation from unity by some optimization technique, as done in [7] using a quintic spline. Instead, we put forward a simpler, more geometric alternative, based on an osculatory Hermite interpolant matching the following conditions at the endpoints 0,1 (Fig. 2a):

1. Customary G^2 data: position, tangent and curvature κ of the original curve.

2. Local arc-length behaviour: unit speed (1) and, in addition, vanishing tangential acceleration.

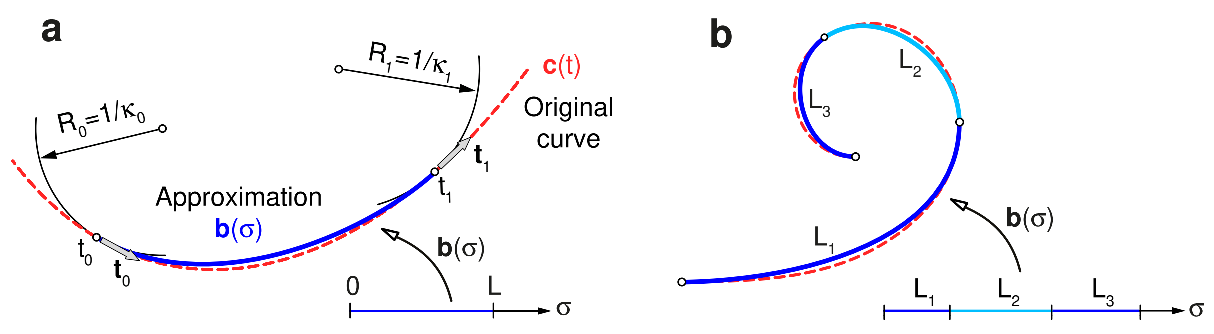


Figure 2: (a) Osculatory Hermite conditions. (b) Subdividing the initial curve $\mathbf{c}(t)$ into several pieces.

If the approximation with a single interpolant is not satisfactory, just subdivide the initial segment $\mathbf{c}(t), t \in [t_0, t_1]$, into k pieces (Fig. 2b), of lengths L_k. Then, for each piece construct its correspondent Hermite interpolant, and concatenate them. The pair of conditions above guarantee that the final piecewise approximation is not only G^2 but also C^2.

3 Polynomial interpolants

3.1 Bézier quintic

In a previous work [6], we already presented the unique Bézier quintic $\mathbf{b}(\sigma)$ that matches the osculatory Hermite conditions posed in the preceding section. This quintic admits a startlingly simple geometric characterization (Fig. 3):

1. Control points $\mathbf{b}_0, \mathbf{b}_1, \mathbf{b}_2$ with projections $\mathbf{b}_0, \mathbf{b}_1, \mathbf{b}_2^t$, along the tangent line $\mathcal{L}$ at $\mathbf{b}_0$, regularly spaced, by a distance L. Equivalently, they display locally a non-parametric geometry, taking $\mathcal{L}$ as abscissa axis.

2. A point $\mathbf{b}_2$ at a signed distance h from $\mathcal{L}$ given by:

$$h = \frac{\kappa_0 L^2}{20}. \qquad (2)$$

Similar relationships hold for the points $\mathbf{b}_5, \mathbf{b}_4, \mathbf{b}_3$, now using the tangent line at $\mathbf{b}_5$.

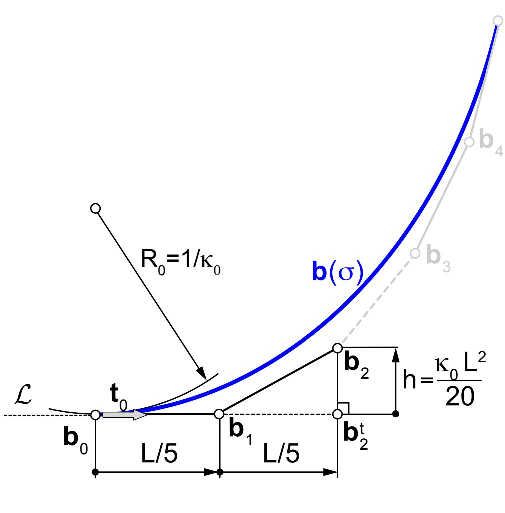


Figure 3: Bézier quintic interpolant: arrangement of the control points $\mathbf{b}_k$.

3.2 Cubic B-spline with two internal knots

Rather than a quintic, to solve our Hermite interpolation we could employ lower-degree cubics, albeit in a piecewise manner. Since some G-code firmwares admit Bézier cubics (G5), we also avoid the loss of accuracy incurred by piecewise linear/circular code [2] required for approximating a quintic.

Using a B-spline representation, 6 control points are required (as in the Bézier case), which means a non-periodic knot vector σ with two internal knots:

$$\sigma = \{0, 0, 0, u_3, u_4, 1, 1, 1\}L. \qquad (3)$$

The cubic B-spline curve $\mathbf{d}(\sigma)$ turns out to be characterized in a similar way to the Bézier case, in terms of distances d_k along the tangent line $\mathcal{L}$ at the endpoint $\mathbf{d}_0$ (Fig. 4):

1. De Boor points $\mathbf{d}_0, \mathbf{d}_1, \mathbf{d}_2$ that display locally a non-parametric geometry. This property implies that:

$$d_1 = u_3\frac{L}{3},\; d_4 = (1-u_4)\frac{L}{3}, \quad d_1 + d_3 = d_2 + d_4 = \frac{L}{3}. \; (4)$$

2. A point $\mathbf{d}_2$ at a signed distance h from $\mathcal{L}$ given by:

$$h = \frac{2}{3}\kappa_0 d_1 d_2,$$

Analogous formulas apply for $\mathbf{d}_5, \mathbf{d}_4, \mathbf{d}_3$

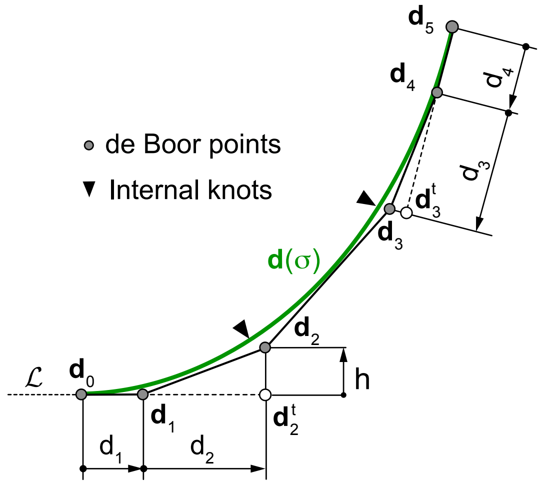


Figure 4: Cubic B-spline interpolant: arrangement of the de Boor points $\mathbf{d}_k$.

Therefore, the construction enjoys two degrees of freedom, namely the two values u_3, u_4 (3), which determine the lengths d_1, d_4 (4), and hence d_2, d_3. These values could be used to optimize the approximation.

4 Free-form design

The schemes of the preceding section also offer potential for free-form design of quasi arc-length curves, rather than approximating a given curve, which makes sense for some applications mentioned in the introduction, such as skinning. To increase design flexibility, we could concatenate several segments in C^2 fashion, simply by setting equal curvatures at the joints.

For each segment, the control points are constrained to meet the quasi-arc length conditions. In the Bézier case, we could set freely:

- The endpoints $\mathbf{b}_0, \mathbf{b}_5$.
- The tangent lines at the endpoints, by choosing $\mathbf{b}_1, \mathbf{b}_4$, with the constrain $|\mathbf{b}_0\mathbf{b}_1| = |\mathbf{b}_4\mathbf{b}_5|$. The position of either $\mathbf{b}_1, \mathbf{b}_4$ determines the constant L.
- The distance h between $\mathbf{b}_2$ and $\mathcal{L}$, which controls the curvature at the endpoint $\mathbf{b}_0$, and analogously for $\mathbf{b}_3$.

In the cubic B-spline case, the inner knot values u_3, u_4 (3) provide two additional degrees of freedom. Rather than setting these knots directly, we could utilize this additional flexibility to choose arbitrary locations for $\mathbf{d}_1, \mathbf{d}_4$. Indeed, these points determine the distances d_1, d_3, and then u_3, u_4 from (4). Research on this subject is under way.

References

[1] R.J. Cripps and P.S. Lockyer. Circle approximation for CADCAM using orthogonal C^2 cubic splines. *International Journal of Machine Tools and Manufacture*, 45:1222–1229, 2005.

[2] R. T. Farouki and K. Hormann. Geometric design: New trends and challenges. *SIAM News*, 52:8, 2019.

[3] R.T. Farouki and T.K. Sakkalis. Real rational curves are not unit speed. *Comput Aided Geom Des*, 8:151–7, 1991.

[4] L. Piegl and W. Tiller. *The NURBS book*. Springer, New York, 2nd edition, 1997.

[5] L. Piegl and W. Tiller. Circle approximation using integral B-splines. *Comput Aided Des*, 35:601–7, 2003.

[6] J. Sánchez-Reyes and J.M. Chacón. A polynomial Hermite interpolant for C^2 quasi arc-length approximation. *Comput Aided Des*, 62:218–226, 2015.

[7] F.C. Wang, B.A. Barsky, D.C.H. Yang, and P.K. Wright. Approximately arc-length parametrized C^3 quintic interpolatory splines. *J Mech Des*, 121:430–9, 1999.

Two-scale mechanical design with effective material property envelope

Xingchen Liu

International Computer Science Institute

Abstract

The adoption of architected and other types of material structures in mechanical design has seen a steady increase due to their abilities to achieve a wide range of material properties and accommodate multi-functional requirements with a single base material. To facilitate the design of multiscale structures with such materials, we propose a novel material property envelope (MPE) that encapsulates the attainable effective material properties of a given material structure family. The MPE interfaces the coarse and fine scales by constraining the combinations of the competing material properties (e.g., volume fraction, Young's modulus, and Poisson's ratio of isotropic materials) during the design of coarse scale material properties. Due to the general lacking of analytical relationships between the competing material properties, we propose a sampling and reconstruction approach to represent the MPE of a given family of structures with the method of moving least squares. The proposed approach enables the analytical derivatives of the MPE, which allows the problem to be solved more accurately and efficiently during the design optimization of the coarse scale effective material property field. The novel interface enables multiscale designs by decoupling design parameters of fine-scale material structures from the coarse scale design and optimization, significantly reducing the complexity while improving the scalability of multiscale design problems. Besides, different material structures, as well as classical engineering materials, can be compared and used concurrently through the proposed envelope, enabling the interchangeable design among multiple families of material structures.

1 Introduction

Additive manufacturing (AM) allows mechanical components with complex shapes and internal structures to be manufactured without significantly increasing the cost or turnaround time. This unique feature of AM leads to a design space that is too vast to be represented by existing design methods. For the first time in an extended period, our ability to design falls short to our ability to manufacture. One effective way to approach such complexity is through scales. Liu and Shapiro [4] proposed a framework for modeling multiscale material structures by the recursive composition of two-scale material structures. The framework links the scales by establishing an explicit relationship between shape–material properties at the fine scale and material properties at the coarse scale and ensures the interchangeability and interoperability of different representations on different scales via queries. Albeit successful in generalizing the existing two-scale modeling approaches through the common mathematical model, the framework is not yet complete for designing multiscale structures in that it lacks a mechanism to constrain the design of the material property on the coarse scale (Figure 1).

In the current work, we propose a novel concept of material property envelope (MPE) that encapsulates all attainable effective material properties of interest by a given family of material structures. The MPE constrains combinations of the competing material properties, e.g., volume fraction vs. Young's modulus, during the design of coarse scale material properties, decoupling design parameters of fine-scale material structures from the coarse scale design and optimization, significantly reducing the complexity which improving the scalability of multiscale design problems. We propose a sampling and reconstruction approach to modeling the MPE of a given family of material structures. In particular, we extend the method of moving least square (MLS) into dimension higher than three to support the high dimension material property envelope. We discuss different cases of reconstruction depending on the relative dimensionality of the material property envelope and material property space. The method of moving least square is used for its ability to provide analytical derivatives of the level set function representing the MPE. This is important because coarse scale material property design is often formulated as an optimization problem, and analytical derivatives allow faster convergence of the optimization, especial with the high dimensional design space. We formulate the problem of coarse scale material property design as an MPE constrained material optimization and demonstrate the effectiveness of the proposed approach with examples and discuss the relationship between the proposed approach and SIMP-based topology optimization.

2 Material property envelope

Let E_i representing the homogenization process for a set of $n \geq 2$ competing material properties $\{p_1, p_2, \cdots, p_n\}$ spaning the material property space P. The material property envelope of a downscaling function D, which is the given process of generating the fine scale structure, can be defined set-theoretically as:

$$\{(p_1, p_2, \cdots, p_n) \in P | \, \forall i, \, \exists S \in im(D) \text{ s.t. } E_i(S) = p_i\} \quad (2.1)$$

where $im(D)$ represents the resultant structure of the downscaling function D.

Apart from the degenerate cases, the dimension of the envelops generally equals the number of design parameters of the structure or the number of the material properties of interest, whichever is less. For example, octet trusses studied in [6] are parametrized by the diameter of the rods in the truss and the Poisson's ratio of the based material. The cubic symmetry in the effective material properties of octet trusses allows the elasticity tensor to be characterized by three material parameters, commonly as Young's modulus E, Shear modulus G, and Poisson's ratio ν (see Figure 2). Together with the volume fraction of the trusses, the MPE of octet trusses resides in a four-dimensional material property space. However, the effective material properties of the trusses will be concentrated around a two-dimensional surface because only two parameters are used to change the shape and material of the trusses. Knowing the dimension of the MPE *a priori* helps the reconstruction of MPE from sampling data set, which are discussed

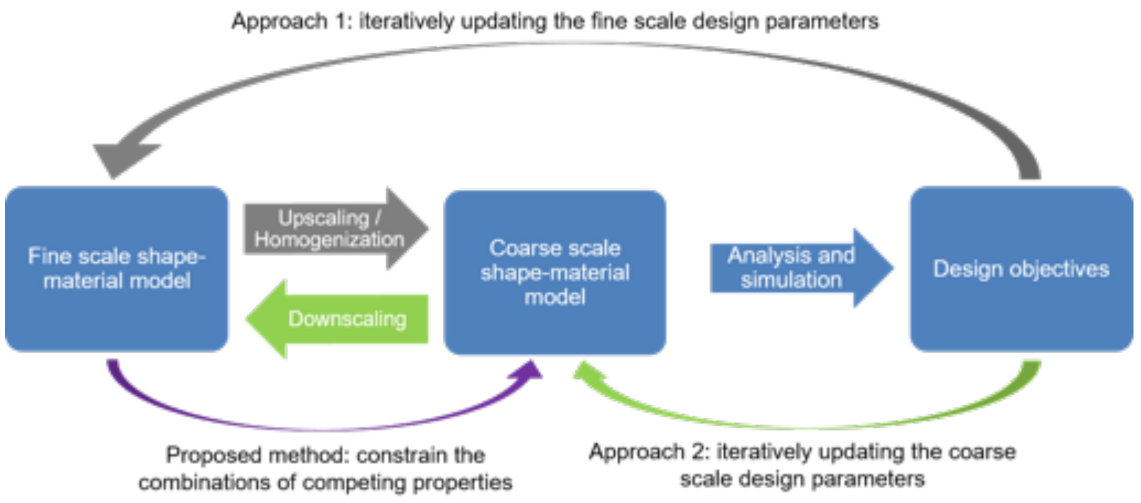

Approach 1: iteratively updating the fine scale design parameters
Fine scale shape-material model
Upscaling / Homogenization
Downscaling
Coarse scale shape-material model
Analysis and simulation
Design objectives
Proposed method: constrain the combinations of competing properties
Approach 2: iteratively updating the coarse scale design parameters

Figure 1: Two approaches of multiscale structure design.

in the following sections. When such information is not readily available, a quick localized principal component analysis (PCA) could help determine the dimension of data set as well.

3 Reconstrcting MPE from sampled material properties

Given a material structure, computing its effective material properties usually involves solving a number of partial differential equations with different boundary conditions [5, 1]. As a result, a direct analytical relationship between competing material properties is generally not available. In the current work, we adopt a sampling and reconstruction approach to approximate the material property envelope defined in Equation 2.1. For practical purposes, the MPE is represented as a level set, which is a scalar field defined over the entire material property space, with a one-to-one correspondence to the set-theoretic definition. The zero and negative level sets correspond to the boundary and the interior of the MPE, respectively.

To approximate the material property envelope, we first populate the space of competing material properties by sampling the effective material properties of the given material structure family. For material structures with procedural or parametrical representations, the material property envelope can be computed through a parameter sweep, where every design parameters of the material structure are sampled at regular or irregular intervals covering the entire range of parameters.

We use the method of moving least squares (MLS) to reconstruct the MPE from the sampling data. The method not only is easy to extend to and scales well with high spatial dimensions to accommodate the high dimensional material property space but also provides analytical derivatives w.r.t every material property of the fitted function with correctly chose weighted function. MLS can be understood through the weighted least squares approximation which adds a local weight function θ to the traditional least square method:

$$\min_f \sum_i \theta(\|\bar{x} - x_i\|)\|f(x_i) - f_i\|^2 \tag{3.2}$$

where f_i is the function value sampled at point x_i, f is a degree m polynomials in d dimensional space:

$$f(x) = b(x)^T c(\bar{x}) \tag{3.3}$$

where $b(x) = [b_1(x), ..., b_k(x)]^T$ is the polynomial basis vector and $c(\bar{x}) = [c_1(\bar{x}), ..., c_k(\bar{x})]^T$ is the vector of unknown coefficients. If θ is locally defined, $f(x)$ is also defined only locally. The unknown coefficients $c(\bar{x})$ is a function of $\bar{x}$ since Equation 3.2 is weighted by distance to $\bar{x}$. By setting the partial derivatives of equation 3.2 w.r.t. $c(\bar{x})$ to zero, $c(\bar{x})$ is solved as

$$c(\bar{x}) = [\sum_i \theta(d_i)b(x_i)b(x_i)^T]^{-1} \sum_i \theta(d_i)b(x_i)f_i \tag{3.4}$$

where $d_i = \|\bar{x} - x_i\|)$

The MLS method was proposed by Lancaster and Salkauskas [2] for surface generation from 3D point cloud data. In MLS, $\bar{x}$ is moved over the entire domain and $f(x)$ can be computed for every $\bar{x}$ location. It has been shown that the global function $f(x)$ is continuously differentiable if and only if the weighting function is continuously differentiable [3]. We use a Gaussian function in the current work. MLS is also a local approach: the interpolated value of any point can be computed locally on demand without processing. This is in stark contrast with the distance field approach [7], which is a global approach and require a significant computational resource to construct.

4 Coarse scale design with MPE

In this section, we formulate the problem of designing coarse scale material property as a PDE-constrained optimization problem which usually takes the form

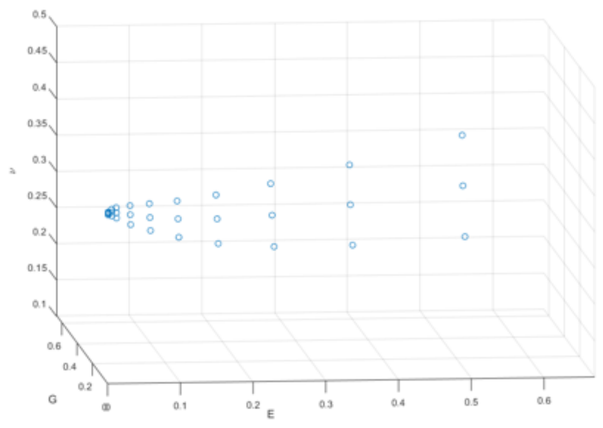
$$\begin{aligned} &\min_m J(u, m) \\ &\text{subject to:} \quad F(u, m) = 0, \quad h(m) = 0, \quad g(m) \leq 0 \end{aligned} \tag{4.5}$$

where the vector m contains the material properties, $F(u, m) = 0$ is a system of PDEs parametrized by m with solution vector u, and $J(u, m)$ is the scalar-valued objective functional that is to be minimised. The equality and inequality constraints $h(m) = 0$ and $g(m) \leq 0$ enforce additional conditions on the material properties. The displacement field u of a structure defined within a fixed design domain Ω is governed by the equations of static equilibrium:

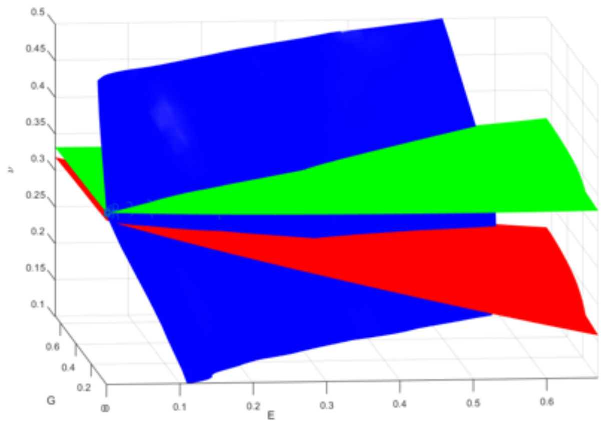
$$\text{div}\sigma + b = 0, \quad x \in \Omega \tag{4.6}$$

under the assumption of small strains

$$\epsilon = \frac{1}{2}[\nabla u + (\nabla u)^T] \tag{4.7}$$



(a) Effective material property samples with varying Poisson's ratio of base material.



(b) Material property envelope as the bounded fitted surface by MLS.

Figure 2: Material property samples and envelopes of Octet truss. Only three dimensions of the four dimensional material space is shown in the plot. The sampling data and polynomial reconstruction are reproduced from [6].

and linear elastic material response

$$\sigma = C\epsilon \tag{4.8}$$

The constitutive relation C is a function of material properties vector m. It is possible that only a subset of m is used to parameterize C as m may include properties, such as volume fraction and surface area, which are not part of the constitutive relation.

Depending on the dimensionality, the MPE may take the form of either equality or inequality constraints. The full dimensional MPE is represented by a level set of negative values, therefore can be modeled as the inequality constraint $g(m) \leq 0$. On the other hand, lower dimensional MPE is represented by the zero level set, therefore can be modeled as the equality constraint $h(m) = 0$. We note that MPE constrains the combinations of material properties such that it is attainable by the given family of material structures as all locations $x \in \Omega$. This constraint will be enforced over every element in the finite element discretization.

Many algorithms for constrained nonlinear optimization require gradients information of the objective function and constraints. Ordinarily, the gradient information of the constraints is calculated numerically by finite difference approximation. In the next section, we provide the partial derivatives of the constraints analytically, allowing the problem to be solved more accurately and efficiently. This is particularly important as the number of constraints we have is on the same order as the number of finite elements. The derivation of the analytical gradient of the objective function is problem dependent and will be explained through examples.

4.1 Analytical partial derivatives of MPE as constraints

When reconstructed by the method of moving least squares, the scalar value representing the MPE is continuously differentiable with a continuously differentiable weighting function θ. Subsituting Equation 3.4 into Equation 3.3, the partial derivative of $f(x)$ w.r.t. x_i:

$$\frac{\partial f(x)}{\partial x_i} = (\frac{\partial b(x)^T}{\partial x_i} A^{-1}B - A^{-1}\frac{\partial A}{\partial x_i}A^{-1}B + b(x)^T A^{-1}\frac{\partial B}{\partial x_i})f_i \tag{4.9}$$

where

$$\begin{aligned} A &= \sum_j \theta(d_j) b(x_j) b(x_j)^T, & \frac{\partial A}{\partial x_i} &= \sum_j \frac{\partial \theta(d_j)}{\partial x_i} b(x_j) b(x_j)^T, \\ B &= \sum_j \theta(d_j) b(x_j), & \frac{\partial B}{\partial x_i} &= \sum_j \frac{\partial \theta(d_j)}{\partial x_i} b(x_j), \end{aligned} \tag{4.10}$$

and j is the index of the sampling data points.

4.2 Sensitivity of the objective function

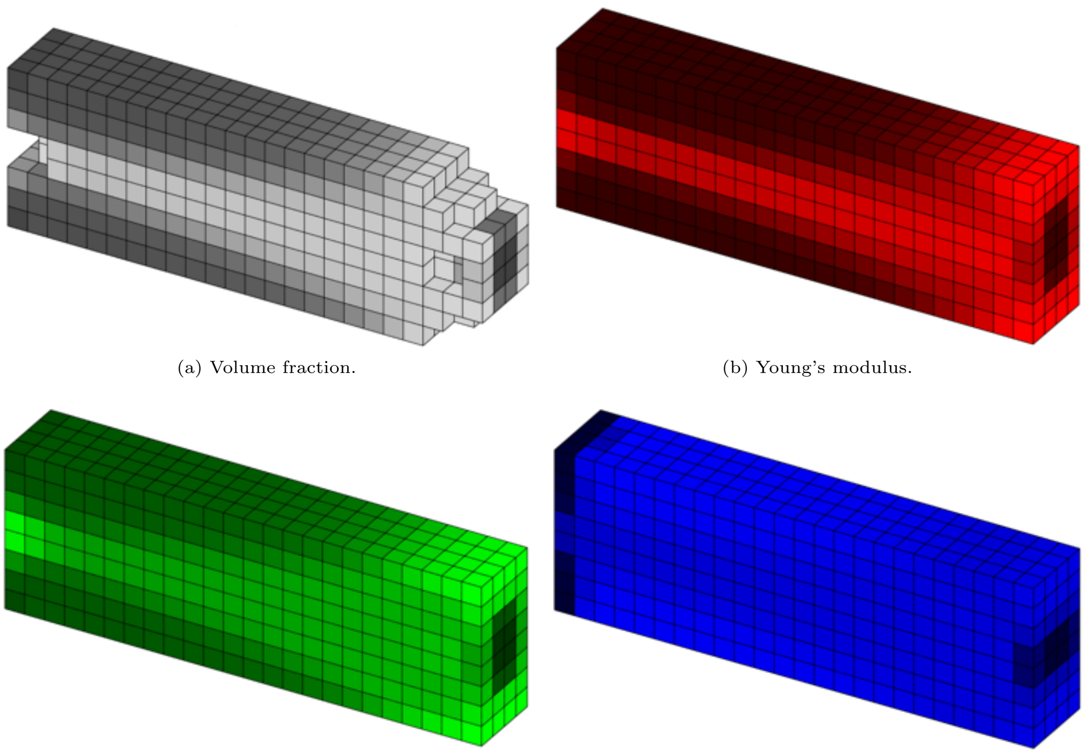
Common objectives of structural design problems include minimizing the compliance or maximize the stiffness of the structure, minimizing the maximum stress within the restructure, matching the deformation profile of the structure to design a compliant mechanism, to list a few. In this section, we focus on the problem of designing the material property field minimizing the compliance of a structure. The designed material property field will be realized through octet truss lattices, whose effective elasticity tensors exhibit cubic symmetry and can be adequately described by Young's modulus E, shear modulus G, and Poisson's ratio ν. Also, the design is constrained by the total volume of the material. The elasticity tensor with cubic symmetry takes the form:

$$C = \begin{bmatrix} C_{11} & C_{12} & C_{12} & 0 & 0 & 0 \\ C_{12} & C_{11} & C_{12} & 0 & 0 & 0 \\ C_{12} & C_{12} & C_{11} & 0 & 0 & 0 \\ 0 & 0 & 0 & C_{33} & 0 & 0 \\ 0 & 0 & 0 & 0 & C_{33} & 0 \\ 0 & 0 & 0 & 0 & 0 & C_{33} \end{bmatrix}, \tag{4.11}$$

where

$$C_{11} = \frac{E(1-\nu)}{(1+\nu)(1-2\nu)}, \quad C_{12} = \frac{E\nu}{(1+\nu)(1-2\nu)}, \quad C_{33} = G \tag{4.12}$$

It is straightforward to derive $\frac{\partial(C)}{\partial(E)}$, $\frac{\partial(C)}{\partial(G)}$, and $\frac{\partial(C)}{\partial(\nu)}$. We note that the partial derivative w.r.t to volumen fraction is zero as C is not a function of the volume fraction of the lattice structure. Figure 3 shows the optimized material property fields of a centilever beam. The designed distribution of effective property will be realized by octet truss lattices.



(a) Volume fraction. (b) Young's modulus. (c) Shear modulus. (d) Poisson's ratio.

Figure 3: Optimization results of coarse scale mateiral property fields of a cantilever beam. Elements with volume fractions less than 20% are not shown in (a).

5 Conclusion

In the present work, we proposed the novel concept of the material property envelope to facilitate the design of multiscale structures with architected materials. The material property envelope encapsulates the attainable effective material properties of a given material structure family and interfaces the coarse and fine scales by constraining the combinations of the competing material properties. A sampling and reconstruction approach is proposed to model the MPE numerically. When used as constraints in design optimization, the MPE is represented as a continuous scalar function over the material property space of interest and provides the analytical partial derivatives to allow the optimization problem to be solved more accurately and efficiently.

By decoupling design parameters of fine-scale material structures from the coarse scale design and optimization, the novel interface significantly reduces the complexity while improving the scalability of multiscale design problems. The proposed formulation provides a mechanism to compare multiple families of material structures as well as classical engineering materials. Multiple material families can work together to enable a large design space through the union of MPEs. An interchangeable design becomes possible through the intersection of MPEs from multiple material families.

References

[1] S. J. Hollister and N Kikuchi. Homogenization theory and digital imaging: A basis for studying the mechanics and design principles of bone tissue. *Biotechnology and bioengineering*, 43(7):586–596, mar 1994.

[2] P. Lancaster and K. Salkauskas. Surfaces generated by moving least squares methods. *Mathematics of Computation*, 1981.

[3] David Levin. The approximation power of moving least-squares. *Mathematics of Computation*, 67(224):1517–1532, 1998.

[4] X. Liu and V. Shapiro. Multiscale shape–material modeling by composition. *CAD Computer Aided Design*, 102, 2018.

[5] Xingchen Liu and Vadim Shapiro. Homogenization of material properties in additively manufactured structures. *Computer-Aided Design*, 78:71–82, may 2016.

[6] Seth Watts and Daniel A White. Simple, accurate surrogate models of the elastic response of three-dimensional open truss micro- architectures with applications to multiscale topology design.

[7] Bo Zhu, Mélina Skouras, Desai Chen, and Wojciech Matusik. Two-Scale Topology Optimization with Microstructures. 36(5), 2017.

Geometrically Smooth Catmull-Clark Spline Surfaces

Ahmed Blidia [1] and Bernard Mourrain [1]

[1] *INRIA Sophia Antipolis Méditéranée*

Abstract

Subdivision schemes such as Catmull-Clark scheme are powerfull tools to produce smooth surfaces that can be easily controlled from a coarse mesh. They became very popular in graphics and animation for their capacities to control easily shapes. However from a geometric modeling point of view, they have some drawbacks: At extraordinary vertices, they are composed of infinitely many rings of piecewise polynomial surfaces and have no explicit analytic representation. Though the limit subdivision surface is smooth, it may not be curvature continuous around an extraordinary vertex [6].

We describe a new explicit scheme to compute a smooth piecewise polynomial surface from a quadrangular meshes. The constructed surface is geometrically smooth everywhere and C^2 except in the neighborhood of extraordinary vertices. The polynomial patches associated to the faces of the quadrangular mesh are bi-quintic Bézier parameterisations. The surface interpolates the Catmull-Clark subdivision surface at the limit points of the vertices of the quad mesh. It has the same tangent plane at these points. Therefore, it is a G^1 approximation of the Catmull-Clark subdivision surface[4]. The Catmull-Clark scheme is used to construct the support of the surface. A degree elevation step and a smoothing step are then applied to obtain the geometrically smooth Catmull-Clark spline surface. These constructions are described explicitly by masks and do not required the solution of linear systems or to solve any optimisation problem.

We also present a new scheme to compute a basis of the space of geometrically smooth functions on the quadrangular mesh. G^0 basis elements are first constructed. The G^1 basis is obtained by a smoothing step. We describe explicit masks to compute these elements from the G^0 elements.

Some recent works propose methods to compute high quality geometrically smooth surfaces over quadrangular meshes. The construction of G^2 surfaces is investigated by solving a constraint minimization problem, using bi-septic patches in [5] or using bi-quintic patches in [3]. In [2], [1], the G^1 surface construction is guided by bi-quintic or rings of bi-quartic Bézier surfaces that minimize some energy. Thus, these constructions involve complex and non-explicit schemes for producing the G^1 surfaces. In our smoothing method, instead of computing smooth guide surfaces, we use the Approximate Catmull-Clark surface as a guide and project it explicitly on the space of G^1 surfaces. This direct and simpler approach provide surfaces of good quality as we will see in the experimentation results.

1 Gluing data

The geometrically smooth constraint corresponds to the following relations: $\forall u \in [0,1]$,

$$\begin{array}{rcl} f_1(u,0) & = & f_0(0,u) \\ a_0(u)\,\frac{\partial f_1}{\partial v}(u,0) & = & a_1(u)\,\frac{\partial f_0}{\partial u}(0,u) + a_2(u)\,\frac{\partial f_0}{\partial v}(0,u) \end{array}$$

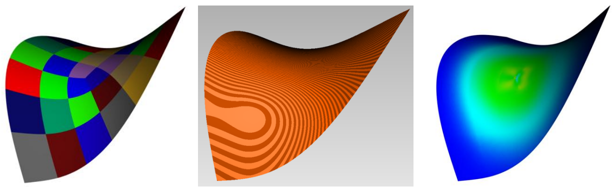


Figure 1: Geometric continuous Catmull-clark surface like a triangular saddle: The left picture represents the surface, in the right the Gauss curvature, and in the left the Isophotes representation

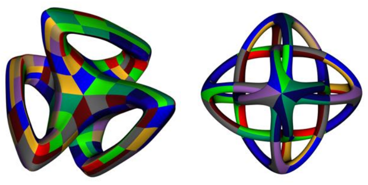


Figure 2: Bi-quintic Bézier patches

where$f_1 = f_{\sigma_1}$, $f_0 = f_{\sigma_0}$ are the restrictions of f on the faces σ_0, σ_1.

we will assume that each singular vertex is isolated from the other singular vertices by at least one layer of ordinary vertices, and we will use the following glueing data along an edge $\tau = (\gamma_0, \gamma_1)$: $a_0(u) = 1$, $a_1(u) = -1$ and

- if $\mathfrak{v}(\gamma_0) \neq 4$ and $\mathfrak{v}(\gamma_1) = 4$, $a_2(u) = \mathfrak{c}(\gamma_0)(1-u)^2$,
- if $\mathfrak{v}(\gamma_0) = 4$ and $\mathfrak{v}(\gamma_1) = 4$, $a_2(u) = 0$.

where $\mathfrak{v}(\gamma)$ is the valence at the vertex γ and $\mathfrak{c}(\gamma) = cos(\frac{2\pi}{\mathfrak{v}(\gamma)})$. the resulting relations between the control points are represented in Fig. 3.

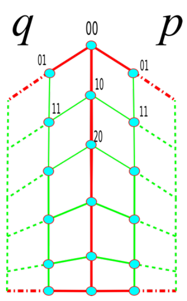
2 G^1 moothing algorithm

After applying the Catmull-Clark algorithm and elevation the degree of the Bézier patches to Bi-5, we project the control points configuration around each singular vertex onto the G^1 constraints in the following way (we use the notation of Fig. 5):

- For each $k \in 1 \ldots \mathfrak{v}$ we compute: $b_{0,1}^{k+1} = \delta_1 b_{0,0}^{k} + \delta_2\, b_{0,1}^{k} - b_{0,1}^{k-1}$ with $\delta_1 = 2 - 2\mathfrak{c}$,$\delta_2 = 2\mathfrak{c}$.
- If the vertex is even for each $k \in 1 \ldots \mathfrak{v}$ we compute:

$$\begin{array}{l} b_{2,0}^{k} = \lambda(-1)^k \sum_{i=1..\mathfrak{v}} (-1)^i b_{1,0}^{i} \\ +(-1)^k \sum_{i=1..\frac{\mathfrak{v}}{2}} (\mu h_{2,0}^{2i+1} + \mu' h_{2,0}^{2i}) \end{array}$$

where $\lambda = \frac{5}{4\mathfrak{v}}(1-\frac{1}{\mathfrak{c}})$, $\mu = (1-\frac{1}{\mathfrak{v}})$, $\mu' = (1+\frac{1}{\mathfrak{v}})$. This guaranties the solvability of the "enclosing constraint", then



$$q_{0,1} = 2\,(1-\mathfrak{c})\,p_{0,0} + 2\,\mathfrak{c}\,p_{1,0} - p_{0,1} \tag{1.1}$$
$$q_{1,1} + p_{1,1} = 2/5\,\mathfrak{c}\,p_{0,0} + 2\,(1-\mathfrak{c})\,p_{1,0} + 8/5\,\mathfrak{c}\,p_{2,0} \tag{1.2}$$
$$q_{2,1} + p_{2,1} = -\,1/5\,\mathfrak{c}\,p_{0,0} + \mathfrak{c}\,p_{1,0} + 2\,(1-\mathfrak{c})\,p_{2,0} + 6/5\,\mathfrak{c}\,p_{3,0} \tag{1.3}$$
$$q_{3,1} + p_{3,1} = 2\,p_{3,0} - 1/5\,\mathfrak{c}\,p_{4,0} + 1/5\,\mathfrak{c}\,p_{5,0} \tag{1.4}$$
$$p_{3,0} = -\,p_{0,0}/10 + p_{1,0}/2 - p_{2,0} - p_{4,0}/2 + p_{5,0}/10 \tag{1.5}$$
$$q_{4,1} + p_{4,1} = 2\,p_{4,0} \tag{1.6}$$
$$q_{5,1} + p_{5,1} = 2\,p_{5,0} \tag{1.7}$$

Figure 3: Relations between the coefficients of two bi-quintic Bézier patches around an edge.

for each $k \in 1 \ldots \mathfrak{v}$ we compute also:

$$\begin{aligned}
b^k_{1,1} &= b^1_{0,0}\mathrm{v}_e + \sum_{i=1..\frac{\mathfrak{v}}{2}} \alpha_i (b^{i+k}_{1,0} + b^{k-i+1}_{1,0}) \\
&+ \sum_{i=1..\frac{\mathfrak{v}}{2}} \beta_i (b^{i+k}_{1,0} + b^{k-i+1}_{1,0}) \\
&+ \gamma_0 h^k_{1,1} + \sum_{i=1..\mathfrak{v}} \gamma_i (h^k_{1,1}) \\
\alpha_k &= (-1)^{k+1}(2-2\mathfrak{c})\frac{\mathfrak{v}-k}{2\mathfrak{v}}, \\
\beta_k &= (-1)^{k+1}\frac{8\mathfrak{c}}{5}\frac{\mathfrak{v}-k}{2\mathfrak{v}}, \\
\gamma_0 &= -\frac{\mathfrak{v}-1}{\mathfrak{v}}, \\
\gamma_k &= (-1)^k \frac{1}{\mathfrak{v}}, \\
\mathrm{v}_e &= \frac{\mathfrak{v}^2 - 2[\frac{\mathfrak{v}}{4}] - 2 + \sum_{i=0..\mathfrak{v}/2}(-1)^i}{2\mathfrak{v}}
\end{aligned}$$

- If the valence is odd,for each $k \in 1 \ldots \mathfrak{v}$ we compute :

  $$\begin{aligned}
  &b^i_{1,1} = \mathrm{v}_0\, b^1_{0,0} \\
  &+ \textstyle\sum_{k=1..i-1}(-1)^{k+i+1}(\theta b^k_{1,0} + \xi b^k_{2,0}) \\
  &+ \textstyle\sum_{k=i..\mathfrak{v}}(-1)^{k+i}(\theta b^k_{1,0} + \xi b^k_{2,0})
  \end{aligned}$$

  where $v_0 = \frac{\mathfrak{c}}{5}$, $\theta = (1-\mathfrak{c})$ and $\xi = \frac{4}{5}\,\mathfrak{c}$.

- For each $k \in 1 \ldots \mathfrak{v}$ we compute: $b^k_{3,0} = \frac{1}{10}\, b^k_{0,0} - \frac{1}{2}\, b^k_{1,0} + b^k_{2,0} + \frac{1}{2}\, b^k_{4,0} - \frac{1}{10}\, b^k_{5,0}$

- For each $k \in 1 \ldots \mathfrak{v}$ we compute:

  $$\begin{aligned}
  b^{k+1}_{2,1} &= \frac{1}{2}h^{k+1}_{2,1} - \frac{1}{2}h^k_{2,1} + s_1\, b^k_{0,0} + s_2\, b^k_{1,0} + s_3 b^k_{2,0} + s_4\, b^k_{3,0} \\
  b^{k+1}_{3,1} &= \frac{1}{2}h^{k+1}_{3,1} - \frac{1}{2}h^k_{3,1} + b^k_{3,0} - s_1\, b^k_{4,0} + s_1\, b^k_{5,0} \\
  s_1 &= \frac{1}{10}\,\mathfrak{c}, s_2 = \frac{1}{2}\mathfrak{c}, s_3 = (1-\mathfrak{c})\,, s_4 = \frac{3}{5}\,\mathfrak{c}
  \end{aligned}$$

- The rest of the constraints are satisfied automatically by the Approximate Catmull-Clark algorithm.

Each one of the steps above is destinated to make sure the control points configuration verify one of the relations in .

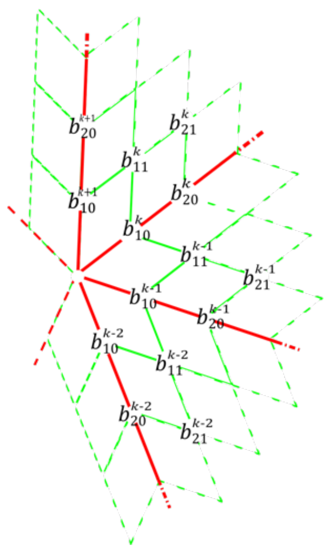


Figure 4: Indices of the b-spline coefficients around a vertex with the convention that $b^k_{i,0} \equiv b^{k-1}_{0,i}$ for $i \in 0 \ldots 5$, $k = 1, \ldots, \mathfrak{v}$.

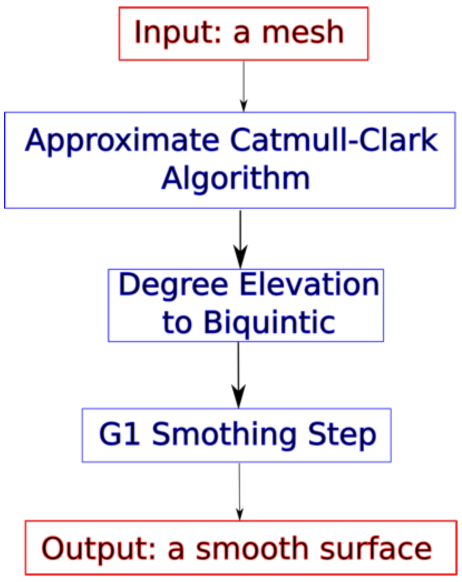

Input: a mesh
Approximate Catmull-Clark Algorithm
Degree Elevation to Biquintic
G1 Smothing Step
Output: a smooth surface

Figure 5: The Global smoothing algorithm from meshes to a smooth surface.

References

[1] Kestutis Karčiauskas and Jörg Peters. Refinable bi-quartics for design and analysis. *Computer-Aided Design*, 102:204–214, 2018.

[2] Ketutis Karčiauskas and Jörg Peters. Improved shape for multi-surface blends. *Graphical Models*, 82:87–98, 2015.

[3] Kestutis Kariauskas and Jörg Peters. Biquintic g2 surfaces via functionals. *Computer Aided Geometric Design*, 33:17 – 29, 2015.

[4] C. Loop and S. Schaefer. Approximating Catmull-Clark subdivision surfaces with bicubic patches. *Acm Transactions on Graphics*, 27(1):11, 2008.

[5] Charles Loop and Scott Schaefer. G 2 Tensor Product Splines over Extraordinary Vertices. 27(5), 2008.

[6] Hartmut Prautzsch. Smoothness of subdivision surfaces at extraordinary points. *Advances in Computational Mathematics*, 9(3):377–389, nov 1998.

3D Reconstruction using 2D triangulation

Florian Gawrilowicz[1] and J. Andreas Bærentzen[1]

[1] *Technical University of Denmark, Department of Applied Mathematics and Computer Science*

Abstract

We present a method for reconstructing a surface from a point set given by multiple range scans. Usually, we think of this surface as a two dimensional manifold embedded in three-dimensional space and the points being samples of this manifold. The key idea of our approach is to construe range scans of a physical object as a set of charts which together form an atlas of the manifold which corresponds to the scanned object. Each chart defines a two-dimensional parametrization of a region of the manifold and thereby of the point samples. This parametrization allows for an efficient triangulation of one region. For the domain of these charts, a natural choice are the image planes of the individual depth images.

1 Introduction

Reconstruction of 3D surfaces from optically acquired data is a big and well-explored field [1]. Overall, methods for 3D reconstruction can be classified as *combinatorial* if the original points are connected or *volumetric* when the surface emerges as the *level-set* of a characteristic function $f : \mathbb{R}^3 \to \mathbb{R}$. The Ball Pivot Algorithm [2] is an example of a combinatorial method whereas Poisson reconstruction [9] is a well-established example of a volumetric method.

Volumetric methods have two features which have traditionally been considered merits: it is easy to ensure that holes (areas with missing points) are closed, and they suppress noise. However, volumetric methods inherently resample the surface since the output mesh is created using a polygonization method such as marching cubes [12]. To accurately capture geometry, we often produce highly detailed meshes, and that can lead to both oversampling and overfitting (see Figure 5).

These are the main reasons we are looking for a combinatorial approach. Our starting point is the observation that the optical acquisition of an object almost always requires scanning it from multiple directions in order to capture the entire surface. In an ideal setting, each scan of the surface would cover a part of the surface that does not overlap that of any other scan, but the opposite is generally true: to ensure proper coverage, scans overlap significantly.

Arguably, this is what makes 3D reconstruction somewhat challenging. Usually, we can reconstruct a surface of a single sub-scan only by 2D triangulation of the points, but then we are left with several partially overlapping triangle meshes. Perhaps, for this reason, most 3D reconstruction methods tend to merge the scans into a single point cloud. We do almost the opposite: our approach is to segment points into disjoint regions where each region corresponds to one of the sub-scans, illustrated in Figure 2. In the image (a) the colors are mixed because the scans overlap, but after segmentation (b) we end up with three point-clouds that cover distinct regions of the surface. Yet each segment corresponds to a sub-scan, and we can reconstruct a surface mesh of the segment simply by 2D triangulating the corresponding point set. What remains is to stitch the seams between each of these meshes, producing the results shown in (e).

1.1 Related Works

Our work may seem similar to the original *zippering* approach [16]. However, zippering removes triangles (along with their vertices) from the boundary of each sub-scan as long as they overlap while we reassign rather than remove points.

In [13] the aim is to construct a global 2D parametrization leading to an optimization problem. In contrast to this, we only have to find the transition functions between the input depth images and show that this is sufficient for 3D reconstruction.

Conceptually, our work is perhaps more closely related to *tangent plane-based triangulation* methods such as [4] which also exploit that surfaces are 2-manifold.

2 Method

Almost invariably, we reconstruct a 3D model from several scans (obtained by optical acquisition) that combine to cover the entire model. In the following, we will use the term *discrete surface map* (DSM) to refer to points in the frame of a single scan. A DSM is a projection of parts of a 3D model onto a 2D domain. Thus, a DSM can be seen both as a discrete chart of the scanned object and as a collection of points for which we have 2D positions in the image domain as well as 3D positions. The domain of each DSM is the corresponding image plane.

We propose to employ the 2D domains of these DSMs as domains for 2D Delaunay triangulation and to perform point cloud reconstruction by subsequently combining these triangulations. Immediately, this seems problematic. It would not lead to a coherent mesh if we triangulated the individual DSMs. The reason being that they overlap, and we would be left with the tedious problem of merging partially overlapping meshes – as illustrated in Figure 1a which shows three scans (blue, orange, and green). Since the three scans are taken from nearby angles, the point clouds almost completely overlap.

In order to approach the problem, we observe that given two DSMs which overlap, say A and B (illustrated Figure 2), it must be possible to map a point $\mathbf{p}$ from the domain of A to the domain of B. In Section 2.3, we propose a method for constructing transition functions that map between pairs of image domains. However, we can also see DSMs as point sets, and the transition functions allow us to reorganize the point sets such that a point which originally belonged to DSM A is mapped to DSM B and then re-assigned to the point set belonging DSM B.

We make use of this by re-assigning points in the overlapping regions to one of the contributing DSMs. Gathering all points sampling a specific patch in one DSM, we reduce the problem to 2D Delaunay triangulation of those patches (see Figure 1b and 1c).

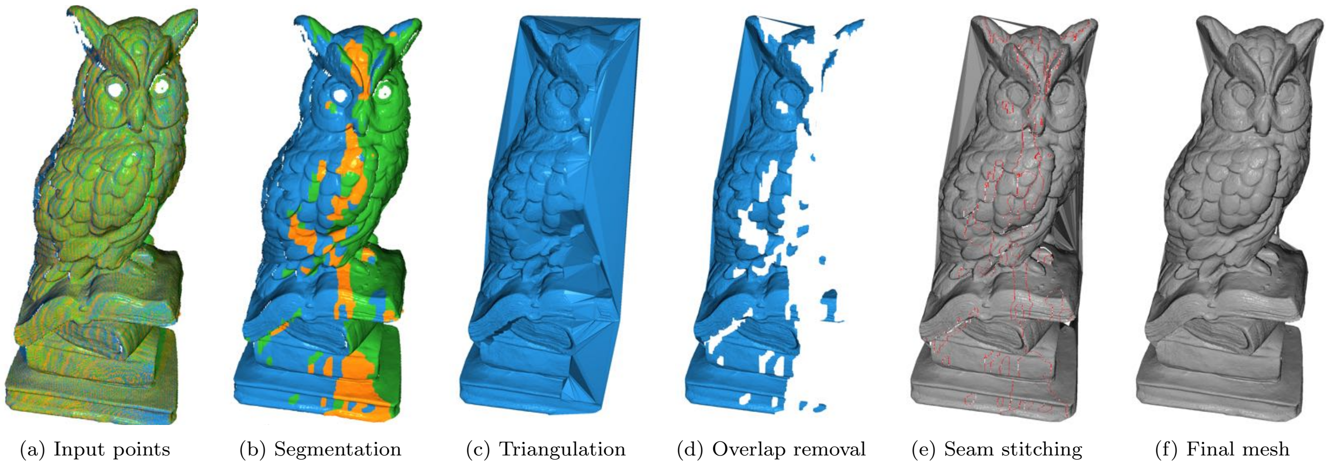


Figure 1: Steps of the algorithm – (a) the different scans are shown in distinct colors, (b) colors represent the assigned image domain, (c) triangulation of partial point cloud assigned to one specific domain, (d) triangulation after removing overlap with other domains, (e) pruned sub-meshes in grey and consensus triangles bridging the gap in red, (f) final result after hole filling.

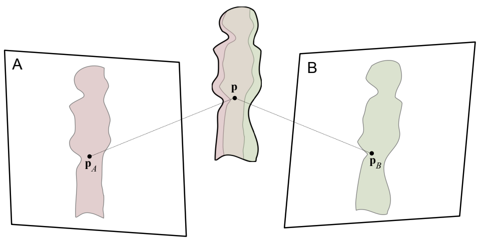


Figure 2: Given a pair of DSMs A and B, we can map a point from its domain position in A, $\mathbf{p}_A$, to its domain position in B, $\mathbf{p}_B$.

2.1 Smoothing

We assume the DSMs are globally and locally aligned, typically using a variant of the ICP algorithm [14]. However, still the surfaces are noisy, and the alignment cannot be assumed to be perfect. Our approach is to establish a (non-meshed) common surface that integrates information from all scans. Therefore we pre-process our data using a method for *mean curvature motion* (MCM) smoothing of point clouds due to [8] – also used in other combinatorial methods like [7, 3]. It is an iterative smoothing process based on the heat equation:

$$\frac{dX}{dt} = h(X)N(X)\,, \tag{2.1}$$

where the position $X \in \mathbb{R}$ evolves in normal direction N proportional to the mean curvature h over time t. It has been shown in [8] that this can be estimated by iteratively projecting each point onto its local total least squares regression plane.

It should be noted that akin to [8] the initial smoothing only serves the purpose of reducing noise for the triangulation and does not introduce smoothing in the final result.

2.2 Segmentation

We need to resort the point sets into non-overlapping pieces. We base our cost function on the information about the origin of each point. For structured light and laser range scanners, this is the scan ID together with the extrinsic parameters. Compared to using normal information it has the following advantages:

- It is resilient to noise as it is given by the scanning process and not derived from noise data.
- It effectively provides us with visibility information such that we do not need to explicitly check for occlusion, which can only be done approximately for a point cloud.

We formulate the segmentation as a graph labeling problem with the points as nodes and connecting the k nearest neighbors with edges. The labels l we assign correspond to the scan IDs and their image planes.

Each point X_i originates from a scan $s(X_i) \in \mathcal{S}$. Our segmentation assigns a label $l_i \in \mathcal{S}$ to each point, which determines the image plane the point is projected into for triangulation. The segmentation minimizes the following energy function via a minimum cut on the neighborhood graph:

$$E = \sum_i \left(u_l(x_i) + \lambda \sum_{j \in \mathcal{N}_i} \mathbb{1}_{l_i \neq l_j} \right)\,, \tag{2.2}$$

where the first term is based on the count of scan IDs $C_l(x_i) = \{x_j | x_j \in \mathcal{N}_i,\, s(x_j) = l\}$ in the neighborhood:

$$u_l(x_i) = 1 - \frac{C_l(x_i)}{\max_m C_m(x_i)} \tag{2.3}$$

and the second term assigns a penalty for each point that gets assigned a label differing from its original scan ID.

Finding the solution to this multi-label problem is done via a minimum graph cut with alpha-expansion [10]. The results are shown in Figure 1b and 3, where we observe that the colors are now separated into contiguous regions. Perceived as charts, the DSMs are unchanged, but the point sets associated with each DSM no longer overlap with points sets associated with other DSMs.

2.3 Transition functions

After assigning each point to a specific DSM, we need to find its 2D position. In order to be more resilient to noise, we do not merely project each point. For each image domain, we construct an auxiliary 2D Delaunay triangulation of the points originating from the corresponding image. By that, we are creating a coarse approximation of the manifold in the embedding space and a corresponding parametrization of the image plane.

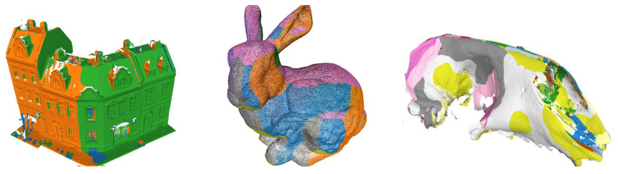


Figure 3: Segmentation into individual domains

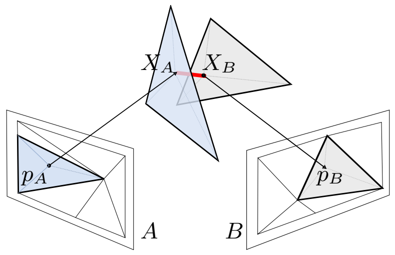


Figure 4: Going from image domain A to B via the connection of X_A and X_B (depicted in red) in embedding space.

In Figure 4 a triangulation of the points in image domain A and B is sketched in conjunction with two triangles of the surrogate surface resulting from that. When mapping a point p_A from A to B, we determine the barycentric coordinates in A. From those, we can compute a position X_A on the surrogate surface in 3D. The closest point to X_A on the surrogate surface corresponding to B denoted as X_B in Figure 4 connects the two DSMs. With the barycentric coordinates of X_B we can then determine the final 2D position on B. This procedure defines our transition functions.

2.4 Seam closing

The remaining task is to join the partial meshes. To address this, we add spatially close points to each part of the segmentation seen in Figure 1b, before triangulating them separately (Figure 1c). This allows us to easily prune each of the resulting sub-meshes by only keeping the triangles connecting points initially labeled as belonging to this part. The result of this pruning step is shown in Figure 1d.

From the set of triangles removed by the pruning, we only retain those triangles connecting one or two points of the original part, e.g., the blue one in Figure 1, with one of the other parts. After processing all segments, we exploit that there tends to be significant consensus [15] between the triangulations in these overlap regions. The consensus triangles connecting the segments are added to the mesh (depicted red in fig. 1e). Lastly, the remaining small holes are closed. The final result is illustrated in Figure 1f.

3 Results

We compare our method to other combinatorial methods, namely *SuperCocone* [5], *RobustCocone* [6], *Co3ne* [3], scale space meshing (SSM) [7]. Also we compare to *Screened Poisson Reconstruction* (SPR) [9] as a widely used volumetric method.

	SPR, depth 10	SPR, depth 11	SSM	Ours
Vertices	625k	1.8M	690k	686k
Faces	1.2M	3.6M	1.4M	1.4M

Table 1: Number of vertices and triangles in the reconstructed surface of the *Owl* model.

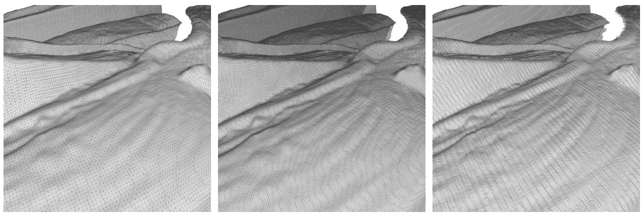


Figure 5: Closeups of the triangulations. From left to right: *Screened Poisson Reconstruction* with maximum octree depth of 10, same with a depth of 11, and our method.

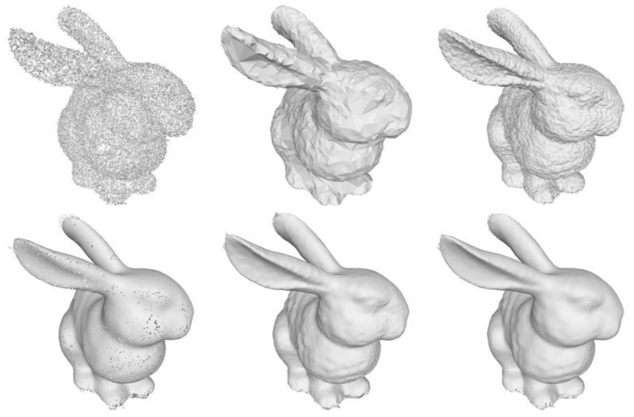


Figure 6: Bunny with added noise of std. 0.3. From left to right: *SuperCocone*, *RobustCocone*, *Screened Poisson Reconstruction* with maximum octree depth of 8, *Scale Space Meshing*, *Co3ne*, and our method.

In Figure 5 and from Table 5 it becomes evident that in order to capture the fine details it is necessary to increase the overall number of points in the final mesh if a volumetric method like SPR is used. This results in an output mesh that models noise along with the details. Our method and *Scale Space Meshing* (SSM) use approximately the same number of points and faces. Whereas *Screened Poisson Reconstruction* (SPR) with a maximum octree depth of 11 uses more than twice as many primitives.

This problem becomes even more apparent when some additional noise is added, which can be seen in Figure 6 and 7. The *SuperCocone* method completely breaks down and is not able to recover a coherent surface at all. *RobustCocone* only uses a fraction of the points and in doing so misses a lot of the details. Poisson reconstruction either has to reduce the resolution or overfits modeling the noise. The ball pivoting step in SSM struggles to include all points leaving holes in the surface. *Co3ne* includes all points but doing so at the expense of topological noise which is very hard to remove with post-processing.

4 Discussion and Future Work

A good case could be made that surface reconstruction from optical scans is a fairly easy problem in the sense that there are many algorithms. However, generally volumetric methods conflate reconstruction and smoothing while also resampling the surface – often leading to an increased number of parameters (vertices) if we aim for the highest level of precision as illustrated in Figure 5. Our approach neither increases the number of points nor does it introduce smoothing.

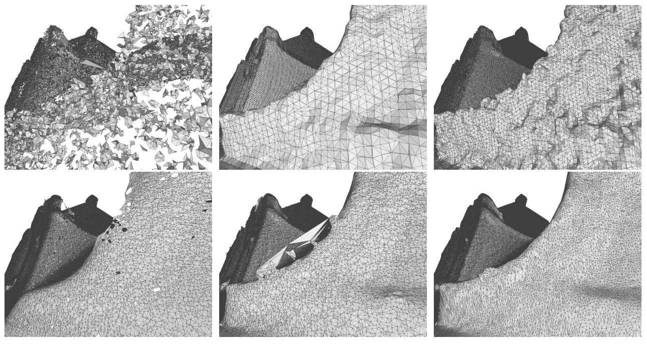


Figure 7: Closeups of the facade scan. From left to right: *SuperCocone*, *Screened Poisson Reconstruction* with maximum octree depth of 10, and 11, *Scale Space Meshing*, *Co3ne*, and our method. (RobustCocone runs out of memory)

Nonetheless there are challenges in the current approach, which we hope to solve in the future:

- When stitching two segments the set of agreeing triangles for difficult camera configurations is rather small, resulting in larger holes in the mesh.
- Although the segmentation is resolving the occlusions well, the fact that we need to add overlaps and restrict inter-segment connections to these, makes deep concavities challenging.
- Thin structures in the geometry pose a problem to the smoothing step, which results either in connections between very dissimilar scan directions or points not being smoothed at all. This problem is also present in [8] and related methods.
- Finding a good balance between the data and the smoothing term for segmentation has been difficult. Shallow minima in the cost function for regions with multiple or no predominant scanning directions require additional regularization. Whereas, too much smoothing causes occluded regions to be falsely assigned.

Also processing times could be reduced at various steps of the algorithm. The graph cut for segmentation only uses a single core although parallel approaches for multi-label segmentation have been proposed [11]. Furthermore, we aim for a rather smooth segmentation and do not depend on the exact location of the borders in between. This would allow us to reduce computational complexity by only using a subset of the points and transferring the label to the whole set afterwards.

Exploring the applicability of our transition function to other problems like mapping color and other quantities derived from 2D images would also be an interesting future direction.

References

[1] Matthew Berger, Andrea Tagliasacchi, Lee M. Seversky, Pierre Alliez, Gaël Guennebaud, Joshua A. Levine, Andrei Sharf, and Claudio T. Silva. A survey of surface reconstruction from point clouds. *Computer Graphics Forum*, 36(1):301–329, Mar 2016.

[2] F. Bernardini, J. Mittleman, H. Rushmeier, C. Silva, and G. Taubin. The ball-pivoting algorithm for surface reconstruction. *IEEE Transactions on Visualization and Computer Graphics*, 5(4):349–359, Oct 1999.

[3] Dobrina Boltcheva and Bruno Lvy. Surface reconstruction by computing restricted voronoi cells in parallel. *Computer-Aided Design*, 90:123 – 134, 2017. SI:SPM2017.

[4] David Cohen-Steiner and Frank Da. A greedy delaunay-based surface reconstruction algorithm. *The Visual Computer*, 20(1):4–16, Apr 2004.

[5] T. K. Dey, J. Giesen, and J. Hudson. Delaunay based shape reconstruction from large data. In *Proceedings IEEE 2001 Symposium on Parallel and Large-Data Visualization and Graphics (Cat. No.01EX520)*, pages 19–146, Oct 2001.

[6] Tamal K. Dey and Samrat Goswami. Provable surface reconstruction from noisy samples. In *Proceedings of the Twentieth Annual Symposium on Computational Geometry*, SCG '04, pages 330–339, New York, NY, USA, 2004. ACM.

[7] Julie Digne. An implementation and parallelization of the scale space meshing algorithm. *Image Processing On Line*, 5:282–295, Nov 2015.

[8] Julie Digne, Jean-Michel Morel, Charyar-Mehdi Souzani, and Claire Lartigue. Scale space meshing of raw data point sets. *Computer Graphics Forum*, 30(6):1630–1642, Feb 2011.

[9] Michael Kazhdan and Hugues Hoppe. Screened poisson surface reconstruction. *ACM Transactions on Graphics*, 32(3):1–13, Jun 2013.

[10] V. Kolmogorov and R. Zabih. Computing visual correspondence with occlusions using graph cuts. *Proceedings of the Ieee International Conference on Computer Vision*, 2:508–515, 2001.

[11] V. Lempitsky, C. Rother, S. Roth, and A. Blake. Fusion moves for markov random field optimization. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(8):1392–1405, Aug 2010.

[12] William E. Lorensen and Harvey E. Cline. Marching cubes: A high resolution 3d surface construction algorithm. *Proceedings of the 14th annual conference on Computer graphics and interactive techniques - SIGGRAPH '87*, 1987.

[13] Nico Pietroni, Marco Tarini, Olga Sorkine, and Denis Zorin. Global parametrization of range image sets. In *ACM Transactions on Graphics (TOG)*, volume 30, page 149. ACM, 2011.

[14] S. Rusinkiewicz and M. Levoy. Efficient variants of the icp algorithm. In *Proceedings Third International Conference on 3-D Digital Imaging and Modeling*, pages 145–152, 2001.

[15] Ryan Schmidt and Patricio Simari. Consensus meshing. *Computers & Graphics*, 36(5):488 – 497, 2012. Shape Modeling International (SMI) Conference 2012.

[16] Greg Turk and Marc Levoy. Zippered polygon meshes from range images. *Proceedings of the 21st annual conference on Computer graphics and interactive techniques - SIGGRAPH '94*, 1994.

CT-shape: Coordinated triangle based reconstruction from dot patterns and boundary samples

Safeer Babu Thayyil [1], Amal Dev Parakkat [2], and Ramanathan Muthuganapathy [1]

[1] *Advanced Geometric Computing Lab., Department of Engineering Design, Indian Institute of Technology Madras, India*
[2] *Computer Science Laboratory of Ecole Polytechnique (LIX), Ecole Polytechnique CNRS, Paris, France*

Abstract

Given a set of points $S \in \mathbb{R}^2$, reconstruction is a process of identifying the boundary edges that best approximates the set of points. In this paper, we propose a unified algorithm for reconstruction that works for both dot patterns as well as boundary samples. The algorithm starts with computing the Delaunay triangulation of the given point set and edges are iteratively removed based on the structure of a pair of triangles. Further, we also propose additional criteria for removing edges based on characterizing a triangle and using degree constraint. Unlike the existing algorithms, the proposed approach requires only a single pass to capture both inner and outer boundaries irrespective of the number of objects/holes. Moreover, the same criterion has been employed for both inner and outer boundary detection. The experiments show that our approach works well for different kinds of inputs. We have done extensive comparisons with state-of-the-art methods for various kinds of point sets including varying the sampling density and distribution and found to perform better or on par with them.

1 Introduction and related works

Given a set of points S lying on a plane, sampled from an object, the reconstruction is a task of embodying the boundary edges (inner and outer boundaries) that best approximates its geometrical identity. In this paper, the point samples are assumed to be derived from a smooth closed curve(s). When the sample points are derived only from the boundaries of the curve(s) (Figure 1(a)), termed as boundary samples, then the reconstruction is generally called as *curve reconstruction* (Figure 1(b)). On the other hand, sample points, in addition to boundaries, can be acquired from the interior to the curve(s) (Figure 1(c)). This sampling is termed as dot pattern and the corresponding reconstruction is called *shape reconstruction* (Figure 1(d)). Devising a unified algorithm that works for both types of input point sets again increases the level of hardness.

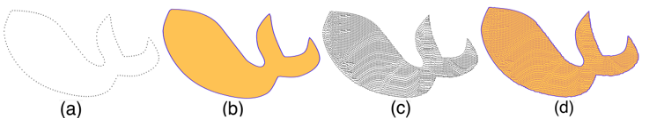


Figure 1: (a) Boundary sample (b) Reconstruction from boundary sample (c) Dot pattern (d) Reconstruction from dot pattern.

Reconstruction algorithms are taxonomised in different ways. It can be based on Delaunay triangulation and non-Delaunay triangulation or curve/shape/unified reconstruction. Some algorithms are designed only for outer boundary detection while others are designed for both outer boundary as well as inner boundary.

Table 1: Strengths and weaknesses of different reconstruction algorithms

Algorithm	Unified	Hole	# Parameters	Multiple Object	Unstructured Hole
α-shape [4]	Y	Y	1	Y	Y
Crust [1]	N	Y	0	Y	Y
nn-crust [2]	N	Y	0	Y	Y
χ-shape [3]	Y	N	1	N	NA
simple-shape [5]	Y	N	3	N	NA
deGoes et. al [6]	N	Y	1	Y	Y
RGG [11]	Y	Y	0	N	N
WDM-crust [12]	N	N	0	Y	N
ec-shape [7]	Y	Y	0	N	Y
HNN-crust [8]	N	Y	0	Y	N
Crawl [10]	N	Y	0	Y	Y
Peel [9]	N	Y	0	Y	Y
Our Algo	Y	Y	0	Y	Y

Table 1 summarizes the strengths and weaknesses of a few of the reconstruction algorithms. Even though there are a lot of works in the area of reconstruction, not all algorithms can handle both types of input - dot patterns and boundary samples. Examples for unified algorithms are α-shape, RGG and ec-shape. Though χ-shape and simple-shape can handle both dot patterns and boundary samples, it can generate only a simple closed curve as output and cannot handle holes. Though unified algorithms such as RGG, *ec*-shape can capture holes, RGG can work only for restricted hole structures and *ec*-shape uses different strategies to capture holes. Crawl and peel can capture different shaped holes but can work only for boundary samples.

α-shape can work for both types of input and independent of the hole structure but requires a parameter α to be tuned. Other approaches such as χ-shape, simple-shape are also parametric algorithms whereas RGG, *ec*-shape, crawl and peel are non-parametric algorithms. It is quite a tedious task to tune the parameter(s) to get the desired output. RGG, *ec*-shape, and χ-shape cannot handle multiple objects.

In this paper, we propose a unified algorithm for the reconstruction of outer boundaries as well as inner boundaries without any user intervention. The following are our **major contributions**: (a) A unified approach for dot pattern and boundary samples with and without holes. (b) The algorithm

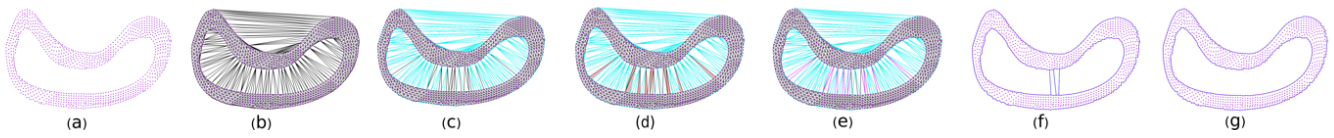


Figure 2: (a) A point set. (b) DT of the point set. (c) marked shared edges of all CT (in cyan). (d) Skinny triangles (in red). (e) marked edges of skinny triangles (in magenta). (f) Graph formed after removing marked edges. (g) Graph after the application of degree constraint.

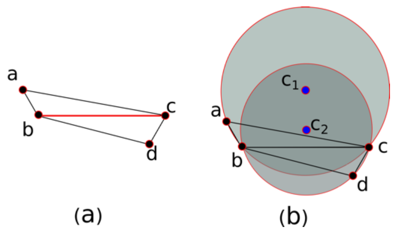


Figure 3: (a) Neighboring triangles $\triangle_{abc}$ & $\triangle_{bdc}$ with shared edge bc. (b) Coordinated triangles $\triangle_{abc}$ & $\triangle_{bdc}$ with circumcenters c_1 & c_2 lying on the same side of the shared edge bc.

uses the same strategy for capturing both hole boundary as well as outer boundary. (c) Our algorithm needs only a single pass irrespective of the number of holes/objects.

2 Definitions

DEFINITION 1 *Neighboring triangles: Two triangles are said to be neighboring triangles if they share an edge (Figure 3(a)).*

DEFINITION 2 *Coordinated triangles: Neighboring triangles are termed as coordinated triangles if their circumcenters lie on the same side of the shared edge (Figure 3(b)).*

DEFINITION 3 *Skinny triangle: A skinny triangle is a thin acute triangle whose base is much smaller than its height.*

DEFINITION 4 *Degree constraint: Only two shorter edges are retained from a vertex (point) and all other edges are removed from that vertex (point).*

3 Algorithm

For a given point set (Figure 2(a)), the algorithm starts with computing the DT of the given point set (Figure 2(b)).

3.1 Marking a shared edge in CT

For each triangle T ∈ DT, the algorithm checks for CT with respect to T. If CT exist, the shared edge between those two triangles is marked (cyan edges in (Figure 2(c)).

3.2 Marking edges from a skinny triangle

Using an angle of 9° for the smallest angle in skinny triangles (shown in red in Figure 2(d)), their two long edges are marked. Figure 2(e) shows all the marked edges so far, in cyan and magenta.

3.3 Applying degree constraint

A graph G is formed from the set of unmarked edges from DT (Figure 2(f)). Since the edges in DT are removed arbitrarily based on CT and skinny triangles, there are possibilities of the presence of non-manifold edges. In order to maintain the output as manifold (for e.g., if there exists only outer boundary, then it should be topologically equivalent to a circle), we impose a degree constraint (Definition 4) on each vertex. Figure 2(g) shows the graph, which is the final reconstructed boundary (in blue) after checking for degree constraint for the point set shown in Figure 2(a).

Algorithm 1 *Complete_Reconstruct*(S)

```
Input: Input point set, $S$.
Output: Reconstructed Output $R$.
1: Construct Delaunay triangulation, $DT(S)$.
2: for each triangle $T$ do
3:     Take all three neighboring triangles and check whether they constitute coordinated triangles.
4:     Mark the shared edges for all coordinated triangles.
5:     Identify the skinny triangles having less than 10° and mark the two longest edges.
6: end for
7: Create a graph $G$ with all unmarked edges of $DT$ (if all three edges are unmarked, they are not considered).
8: Apply degree constraint on all vertices of $G$.
9: return $G$ as CT-shape
```

The pseudo-code for the algorithm for reconstruction, given a set of points S is delineated in Algorithm 1. The running time complexity of the algorithm can be shown to be $O(n \log n)$ where n is the number of points in S.

4 Results & Discussions

Our algorithm (Algorithm 1) is implemented in C++ with CGAL (Version: 4.6) libraries and visualized in OpenGL and tested in MacOS 10.12.3. The input point sets (dot patterns and boundary samples) consist of points from simple objects, objects with multiple holes, objects with multiple components, objects with non-divergent concavities etc. The algorithm has also been tested with different sampling densities and distributions. Figure 4 shows some of the results of our algorithm for various dot patterns and boundary samples. The figure shows that our algorithm can generate good results for both kinds of inputs with divergent features.

4.1 Comparison with existing algorithms

Here we considered five algorithms (α-shape, χ-shape, simple-shape, RGG, ec-shape with ours) for dot pattern and nine algorithms (α-shape, χ-shape, simple-shape, RGG, ec-shape, Crawl, HNN-crust, Peel, WDM-crust with ours) for boundary samples for the sake of comparison. It may be noted that HNN-crust, WDM-crust, Crawl and Peel do not work for dot patterns and hence they have been included only for the comparison of results for boundary samples as input. In all the comparison results, we use circles to denote regions where boundaries are not well-approximated.

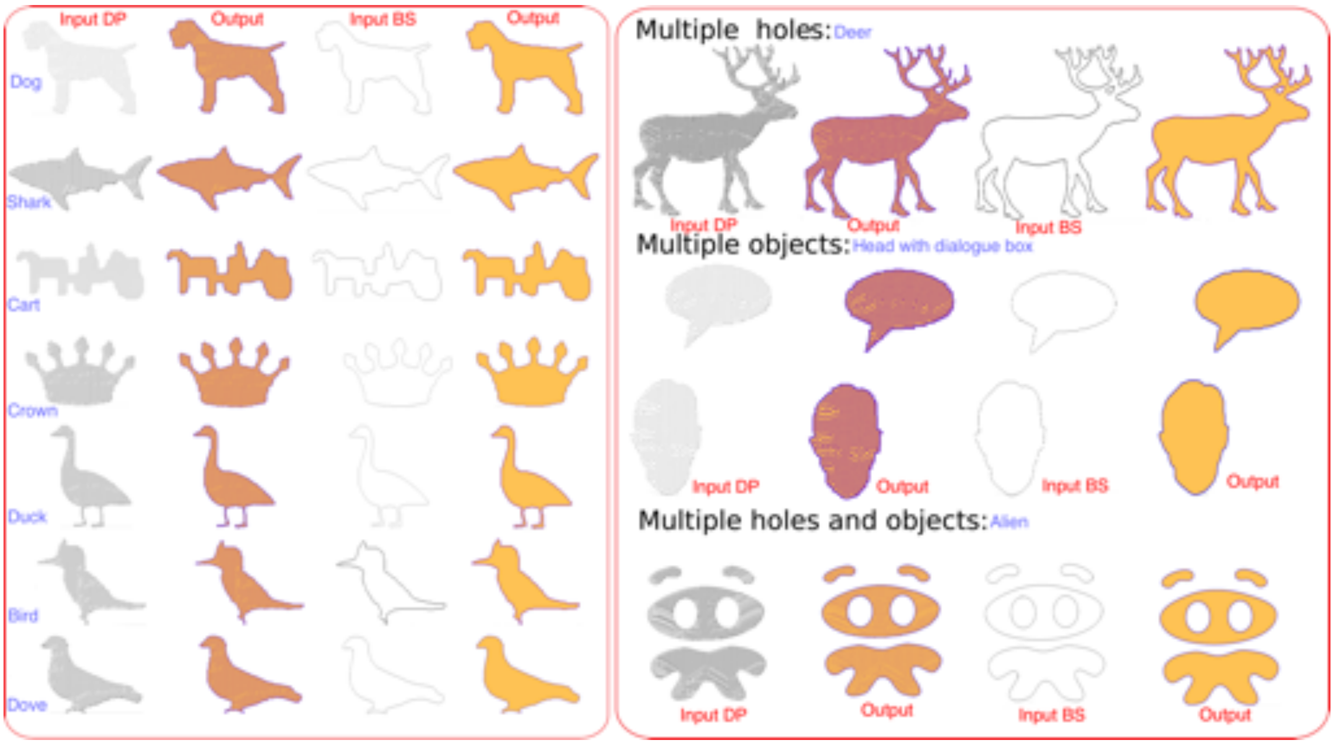


Figure 4: Results of our algorithm (CT-shape) for various features like concavity, multiple holes (deer has two holes), multiple components etc. Input DP = Input dot pattern, Input BS = Input boundary sample, Output = Output of our algorithm.

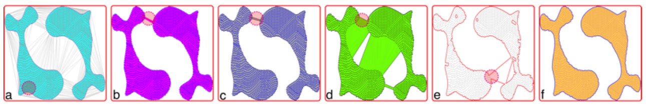


Figure 5: Multiple objects (dot pattern): Results of (a) α-shape (b) χ-shape (c) simple-shape (d) RGG (e) ec-shape (f) our result (CT-shape). Some algorithms (as indicated in circles) have resulted in single object even when multiple objects are present.

4.1.1 Qualitative Comparison

Capturing multiple objects is a challenge for many algorithm as they work only for single component only. Figures 5 and 6 show the comparison results for multiple objects. Figures 7 and 8 show the result of various algorithms for a point set sampled from an object with multiple holes. Our algorithm has captured all the holes reasonably well.

4.1.2 Quantitative Comparison

We made use of L^2-error norm [3] to compare the results, which is defined as (C and P are the ground truth and reconstructed result respectively):

$$L^2 = \frac{area((C-P)\bigcup(P-C))}{area(C)} \tag{4.1}$$

Figure 9 shows the point density versus L^2 error plots of our experimentation on various point sets extracted from the boundary of the country shapes (Paraguay and Spain) for both dot patterns and boundary samples. From Figure 9, it is clear that our algorithm works better or on par with various unified reconstruction algorithms.

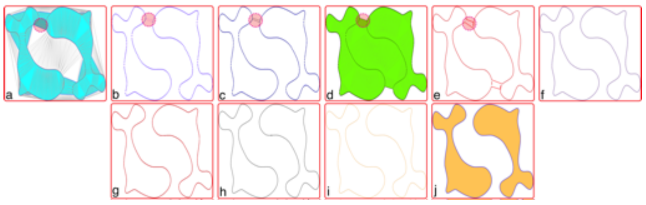


Figure 6: Multiple objects (boundary sample): Results of (a) α-shape (b) χ-shape (c) simple-shape (d) RGG (e) ec-shape (f) Crawl (g) HNN-crust (h) Peel (i) WDM-crust (j) our result (CT-shape). A few of the algorithms (as indicated in circles) have resulted in single object even when multiple objects are present.

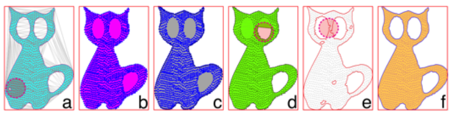


Figure 7: Object with holes (dot pattern): Results of (a) α-shape (b) χ-shape (c) simple-shape (d) RGG (e) ec-shape (f) our result (CT-shape). χ-shape and simple-shape capture only outer boundaries. RGG works only if the holes are body-arm structured. ec-shape overdigs the holes.

4.1.3 Varying point distributions

Figure 10 shows the comparison of our results with other unified algorithms for various point distributions. The four instances of point distributions we used for experimentation are: Dense Boundary Dense Internal (DBDI), Dense Boundary Sparse In-

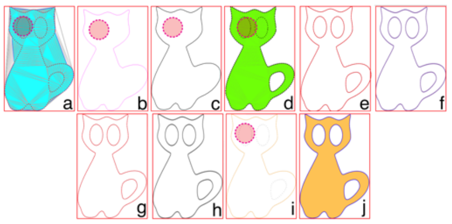


Figure 8: Object with holes (boundary sample): Results of (a) α-shape (b) χ-shape (c) simple-shape (d) RGG (e) ec-shape (f) Crawl (g) HNN-crust (h) Peel (i) WDM-crust (j) our result (CT-shape).

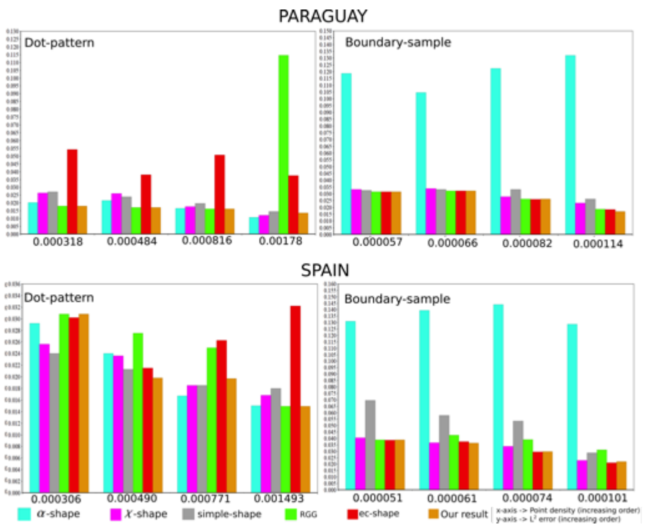


Figure 9: Exemplification of performance comparison on different point densities of different country shapes.

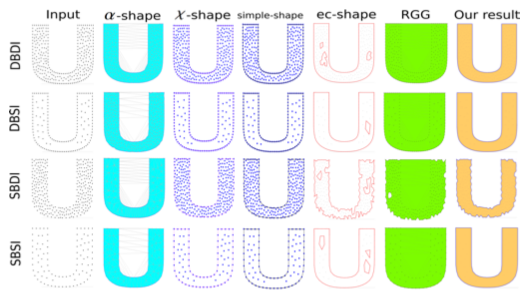


Figure 10: Results showing the performance of different algorithms on various point distributions.

ternal (DBSI), Sparse Boundary Dense Internal (SBDI) and Sparse Boundary Sparse Internal (SBSI). Our algorithm has captured the details quite well except in the case of SPDI.

5 Conclusion

In this paper, we devised a unified reconstruction algorithm and showed it works irrespective of the type of point set (dot pattern or boundary sample). The algorithm is easy to implement and proven to give good results under various point densities and distributions. In contrast to other unified algorithms, our algorithm needs only a single pass to detect the boundaries (both inner and outer) irrespective of the number of holes/objects. As a future work, we would like to extend the algorithm to look into various other challenging tasks like handling point sets with noise and outliers. We are also working on the extension of the proposed algorithm to higher dimensions.

References

[1] Nina Amenta, Marshall Bern, and David Eppstein. The crust and the beta-skeleton: Combinatorial curve reconstruction. In *Graphical Models and Image Processing*, pages 125–135, 1998.

[2] Tamal K. Dey and Piyush Kumar. A simple provable algorithm for curve reconstruction. In *SODA '99*, pages 893–894, 1999.

[3] Matt Duckham, Lars Kulik, Michael F. Worboys, and Antony Galton. Efficient generation of simple polygons for characterizing the shape of a set of points in the plane. *Pattern Recognition*, 41(10):3224–3236, 2008.

[4] Herbert Edelsbrunner, David G. Kirkpatrick, and Raimund Seidel. On the shape of a set of points in the plane. *IEEE Transactions on Information Theory*, 29(4):551–558, 1983.

[5] Amin Gheibi, Mansoor Davoodi, Ahmad Javad, Fatemeh Panahi, Mohammad M Aghdam, Mohammad Asgaripour, and Ali Mohades. Polygonal shape reconstruction in the plane. *IET computer vision*, 5(2):97–106, 2011.

[6] Fernando de Goes, David Cohen-Steiner, Pierre Alliez, and Mathieu Desbrun. An Optimal Transport Approach to Robust Reconstruction and Simplification of 2D Shapes. *Computer Graphics Forum*, pages 1593–1602, 2011.

[7] Subhasree Methirumangalath, Shyam Sundar Kannan, Amal Dev Parakkat, and Ramanathan Muthuganapathy. Hole detection in a planar point set: An empty disk approach. *Computers & Graphics*, 66:124–134, 2017.

[8] Stefan Ohrhallinger, Scott A. Mitchell, and Michael Wimmer. Curve reconstruction with many fewer samples. *Computer Graphics Forum*, 35(5):167–176, 2016.

[9] Amal Dev Parakkat, Subhasree Methirumangalath, and Ramanathan Muthuganapathy. Peeling the longest: A simple generalized curve reconstruction algorithm. *Computers & Graphics*, 2018.

[10] Amal Dev Parakkat and Ramanathan Muthuganapathy. Crawl through Neighbors: A Simple Curve Reconstruction Algorithm. *Computer Graphics Forum*, pages 177–186, 2016.

[11] Jiju Peethambaran and Ramanathan Muthuganapathy. A non-parametric approach to shape reconstruction from planar point sets through Delaunay filtering. *Computer-Aided Design*, 62:164 – 175, 2015.

[12] Jiju Peethambaran, Amal Dev Parakkat, and Ramanathan Muthuganapathy. A Voronoi based labeling approach to curve reconstruction and medial axis approximation, 2015.

Scikit-Shape: Python Toolbox for Shape Analysis and Segmentation

Günay Doğan [1]

[1] e-mail:gunay.dogan@nist.gov
[1] Theiss Research, National Institute of Standards and Technology

1 Introduction

We present scikit-shape, a free open-source Python package for image segmentation and shape analysis, available at http://scikit-shape.org. The package includes image segmentation algorithms to detect distinct regions or objects and their boundaries in given images. It also includes elastic shape distance algorithms to compute dissimilarity scores of curve boundaries for statistical shape analysis. The package builds on NumPy and SciPy libraries, which provide an efficient infrastructure for numerical computations [6]. It also interfaces and interacts well with the image processing package, scikit-image [11], and the machine learning package, scikit-learn [8], so that it can be used as a powerful component of richer and more general image and data analyses. In the following sections, we list and describe the algorithms that are included.

2 Image Segmentation

Image segmentation is the problem of identifying distinct regions or objects and their boundaries in given images. It is a fundamental problem in image processing, and a critical component of many image analysis tasks. Locating cells in microscopy images or decomposing a material microstructure into grains are a few example applications of image segmentation among many others in sciences and engineering. As images and specific instances of segmentation problems show a lot of variability, many different approaches have been developed over the recent decades. Our package includes segmentation algorithms of three different types:
1) shape optimization of region boundaries,
2) evolution of phase field functions representing regions,
3) topology optimization of regions encoded as pixel labels.
All of these three approaches are executed as iterative optimization [2, 3]. An initial guess for segmentation, i.e. a set of curves, a phase field function, or initial assignment of region labels, is set by the user, or by an automatic initialization routine. Then the optimization algorithms update the segmentation iteratively until the optimal configuration is attained (see Fig.1 for an example). This process is guided by one of a set of specially-designed segmentation energies encoding how well a given configuration (of boundary curves or phase field function or region labels) capture the actual regions, objects, or their boundaries in the image. There are several choices for segmentation energies, those based on image edge criterion, on image intensity contrast criterion, or on statistical considerations. The segmentation energy can include terms to control the smoothness of the boundaries and/or statistical priors

3 Image-based Meshing

Some applications require derived measurements based on the image segmentation, and sometimes these derived measurements are obtained by numerical simulations of the physics on a simplicial mesh, i.e. triangulation in 2d, representing the geometry of the structures in the image, based on the segmentation. An example is the OOF software package for finite element analysis of material microstructure physics, for which the starting point is a high quality mesh obtained from the segmentation [7]. To create such meshes, our package interfaces with the TRIANGLE program, which produces high-quality triangulations with theoretical guarantees [9]. These triangulations conform perfectly to the region boundaries that are passed to TRIANGLE by our package. An example of such a triangulation is shown in Fig.2.

4 Shape Analysis

In many applications, one needs to quantitatively compare the shapes of objects using shape dissimilarity or shape distance scores. There are several alternative approaches to computing shape distances. Our approach uses the elastic shape distance framework of Srivastava et al. [10], and the elastic shape distance algorithm that we implement is the foundation of the shape analysis capabilities enabled by our package. This shape distance framework is particularly desirable for applications, as it can handle natural variations of boundary shapes (e.g. all maple leave boundaries are slightly different, but they all have about the same shape). Moreover, the shape distance, as expected, is invariant to scaling, translation, rotation of boundary curves, and can be based on tangents, curvature or angle function of the curves. Once the shape distance is computed, it can be used in conjunction with other tools, such as spectral clustering or kernel density estimation, to perform various statistical analyses. We provide an efficient optimization algorithm to compute the elastic shape distances. This algorithm leverages fast algorithmic components, such as FFT-based curve alignment, fast dynamic programming for curve parameterization, and integrates them in an efficient optimization scheme [1, 4, 5].

References

[1] Javier Bernal, Gunay Dogan, and Charles R Hagwood. Fast dynamic programming for elastic registration of curves. In *Proc. of DIFFCVML Workshop*, pages 111–118, 2016.

[2] Günay Doğan. An efficient curve evolution algorithm for multiphase image segmentation. In *Proc. of EMMCVPR*, pages 292–306. Springer, 2015.

[3] Günay Doğan. An efficient lagrangian algorithm for an anisotropic geodesic active contour model. In *Proc. of SSVM*, pages 408–420. Springer, 2017.

[4] Günay Doğan, Javier Bernal, and Charles R Hagwood. A

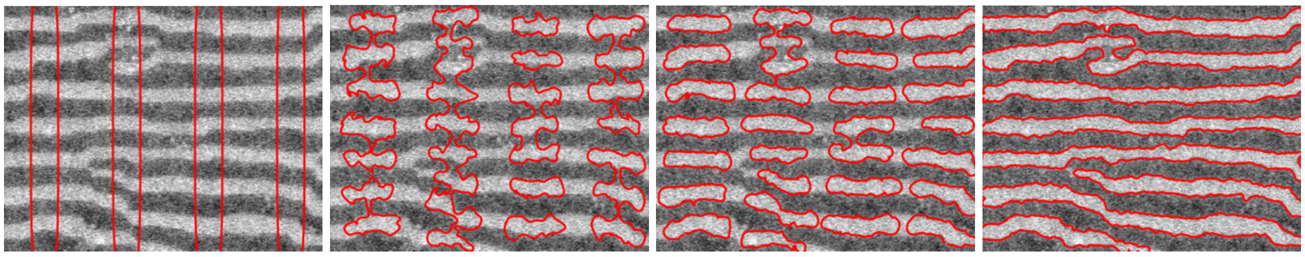


Figure 1: Some snapshots from the shape optimization of region boundary curves for segmentation. The image on the left contains the initial curves. The image on the right is the final segmentation.

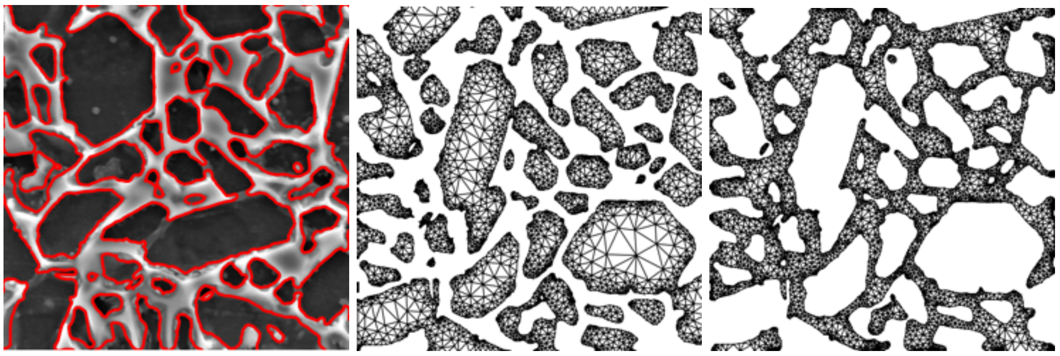


Figure 2: Triangulated meshes of the interior and exterior regions, based on the segmented boundary curves.

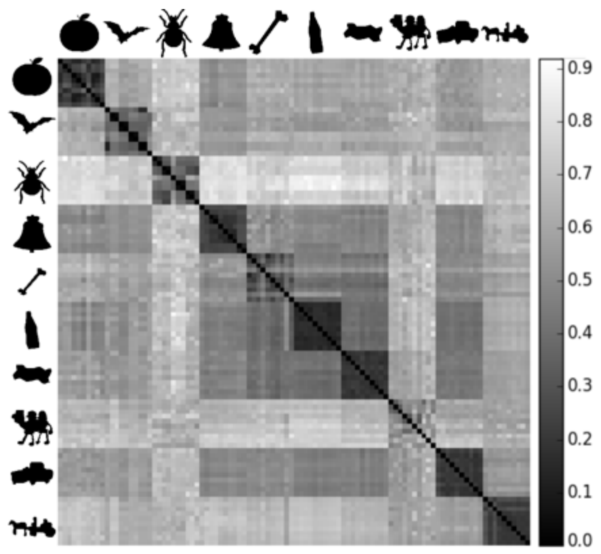


Figure 3: Grayscale visualization of the matrix of shape distances between pairs shape examples from the MPEG7 shape benchmark data set.

fast algorithm for elastic shape distances between closed planar curves. In *Proc. of CVPR*, pages 4222–4230, 2015.

[5] Günay Doğan, Javier Bernal, and Charles R Hagwood. FFT-based alignment of 2d closed curves with application to elastic shape analysis. In *Proc. of DIFFCV Workshop*, pages 4222–4230, 2015.

[6] Eric Jones, Travis Oliphant, Pearu Peterson, et al. SciPy: Open source scientific tools for Python, 2001–. [Online; accessed 2019-04-01].

[7] Stephen A Langer, Edwin R Fuller, and W Craig Carter. OOF: an image-based finite-element analysis of material microstructures. *Computing in Science & Engineering*, 3(3):15–23, 2001.

[8] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. Scikit-learn: Machine learning in Python. *Journal of machine learning research*, 12(Oct):2825–2830, 2011.

[9] Jonathan Richard Shewchuk. Triangle: Engineering a 2D Quality Mesh Generator and Delaunay Triangulator. In *Applied Computational Geometry: Towards Geometric Engineering*, volume 1148, pages 203–222. Springer-Verlag, May 1996.

[10] A. Srivastava, E. Klassen, S.H. Joshi, and I.H. Jermyn. Shape analysis of elastic curves in Euclidean spaces. *PAMI*, 33(7):1415–1428, 2011.

[11] Stefan Van der Walt, Johannes L Schönberger, Juan Nunez-Iglesias, François Boulogne, Joshua D Warner, Neil Yager, Emmanuelle Gouillart, and Tony Yu. scikit-image: image processing in Python. *PeerJ*, 2:e453, 2014.

Variational Shape Approximation of Point Set Surfaces

Martin Skrodzki [1], Eric Zimmermann [1], and Konrad Polthier [1]

[1] *Freie Universität Berlin, Germany*
[1] *martin.skrodzki@fu-berlin.de, eric.zimmermann@fu-berlin.de, konrad.polthier@fu-berlin.de*

Abstract

This work proposes an algorithm for point set segmentation based on the concept of Variational Shape Approximation (VSA), which uses the k-means approach. It iteratively selects seeds, grows flat planar proxy regions according to normal similarity, and updates the proxies. It is known that this algorithm does not converge in general. We provide a concrete example showing that the utilized error measure can indeed grow during the run of the algorithm. To reach convergence, we propose a modification of the original VSA. Further, we provide two new operations applied to the proxy regions, namely split *and* merge, *which enqueue in the pipeline and act according to a user-given parameter. The advantages over regular VSA are independence of both a prescribed number of proxies and a (manual) selection of seeds. Especially the latter is a common drawback of region-growing approaches in segmentation.*

1 Introduction

Point sets arise naturally in almost all kinds of three-dimensional acquisition processes, like 3D laser-scanning and have been recognized over 30 years ago as fundamental shape for representation in computer graphics. In comparison to meshes, they have a decreased demand in storage and have the advantage to be the direct representation of the object as obtained from acquisition devices, whilst lacking connectivity information.

However, in many applications, large parts of the point set carry redundant information. For example, a flat area of a surface can be sampled sparsely compared to an area of high curvature. The identification of such flat areas can be achieved e.g. via segmentation. Cohen-Steiner et al. proposed the Variational Shape Approximation (VSA), [1]. The procedure segments a given mesh into a given number of regions approximated by proxies, which can be used for a simplified model. A translation to point sets was done by Lee et al. [4] with a focus on feature extraction.

In a survey of simple geometric primitives detection methods for captured 3d data, the authors of [3] find VSA to be a method in particular suited to be run on individual algorithms as opposed to in- or outdoor scene data. It is summarized as an "automatic clustering" approach with low abstraction level and medium data fidelity, which attains a good balance in terms of e.g. speed, scalability, simplicity, and generality when compared to other methods, see [3] for details.

Despite its advantages, the VSA procedure and its translations have several downsides. First, as [1] also states, the procedure is not convergent in general, which is the same for meshes and point sets alike. Second, the available variants assume a prescribed number of proxies. Third, the quality of the final segmentation depends on the choice of starting seeds for the proxies. Our main contributions are:

- Provide an example of a growing error during the run of the VSA algorithm which applies to meshes [1] and point sets [4] alike.
- Presentation of a modified VSA version and proof of its convergence.
- Description of two new operations, *split* and *merge*, in the VSA pipeline, making the initial choice of a fixed proxy number and manual seed selection unnecessary.

2 VSA Procedure

The VSA procedure partitions a surface $S \subseteq \mathbb{R}^3$ into $m \in \mathbb{N}$ disjoint regions $R_i \subseteq S$, $\sqcup R_i = S$, where each region is associated a linear proxy $P_i = (C_i, N_i) \in \mathbb{R}^3 \times \mathbb{S}^2$, where C_i denotes the center and N_i denotes an associated unit-length normal, i.e. every proxy appears as a plane. After an initial seed selection every region grows w.r.t. a metric given by

$$\mathcal{L}^{2,1}(R_i, P_i) = \int_{x \in R_i} \|n(x) - N_i\|^2 \, dx,$$

where $n(x)$ denotes the surface normal at point $x \in S$. Throughout the whole paper, with $\|\cdot\|$ we refer to the Euclidean norm. Observe that this is the second proposed metric in [1] and the first one considers only the point positions. We focus on the one driven by normals as the authors found it to be favorable. In the discrete setting, where S is given as a (triangulated) mesh with elements t_j, the error metric simplifies to

$$\mathcal{L}^{2,1}(R_i, P_i) = \sum_{t_j} \|n(t_j) - N_i\|^2 \, |t_j| \,, \tag{2.1}$$

with $n(t_j)$ the element normal and $|t_j|$ its area. In their adaption to point sets, Lee et al. replaced the element normals and area term in Equation (2.1) by vertex normals and weighted all points equally with value 1. Here, it is possible to introduce more complex weighting terms in the point set setting (e.g. [6]), since we do not have an adequate equivalent to the area of an element. Afterwards, in both the mesh and the point set setting, the error measure

$$E(\{(R_i, P_i) \mid i = 1, \ldots, m\}) = \sum_{i=1}^{m} \mathcal{L}^{2,1}(R_i, P_i). \tag{2.2}$$

is minimized.

In order to find a minimum of the above error functional, the VSA procedure relies on a variation of Lloyd's k-means algorithm [5]. It works on both meshes and point sets, while the latter just uses the points, its normals, and a proper notion of neighborhoods. From all respective elements, m are chosen randomly to build up the proxies, with C_i as a proxy's barycenter and N_i its normal. The neighbors of selected elements are collected into a priority queue $\mathcal{Q}$ and sorted increasingly with growing $\mathcal{L}^{2,1}$. Afterwards the following three steps are performed iteratively until convergence:

1. *Flood:* As long as $\mathcal{Q}$ is not empty, pop the first element. Ignore it, if it has already been assigned to a proxy. If not, assign it to the proxy that pushed it into $\mathcal{Q}$ and collect all neighboring elements into the queue with proxy label they got pushed by.
2. *Proxy Update:* Update all proxy normals as averaged sum of the normals of their associated elements.
3. *Seeds:* For each proxy respectively, find an element in each region which is most similar according to the associated proxy normal and use it as seed element for the next flooding step.

In the work of Lee et al. [4], the authors use the k nearest neighbors as their neighborhood notion, with $k \in \{15, \dots, 20\}$.

3 Example for a Growing Error Functional

Although the authors of [1] state that they cannot guarantee global convergence, they do not provide a concrete example. In this work, we contribute to the understanding of the algorithm by describing a setup in which the error function (2.1) does grow during the run of VSA.

Consider the 2-dimensional setup shown in Figure 1(a) with n points given connected on a line with normal $\binom{-1}{1}$ next to a line of n points with normal $\binom{0}{1}$. At the right end of the second line, there is a single point with normal $\binom{-1}{0}$ and another single point with normal given by

$$N = \frac{1}{n+2}\left(n \cdot \begin{pmatrix}0\\1\end{pmatrix} + \begin{pmatrix}-1\\0\end{pmatrix} + N\right).$$

Now, two proxies will act on this example, with their initial seeds shown in yellow and blue in Figure 1(a). They each start on one of the two lines of n points respectively. The result after a flood is shown in Figure 1(b), where each line is completely covered by the proxy starting on it and the two single points are associated to the proxy with normal $\binom{0}{1}$. After updating the proxy normals, the yellow proxy has normal $\binom{-1}{1}$ while the blue proxy has normal N given by the equation above. Thus, the yellow proxy starts from an arbitrary point on its line while the blue proxy starts from the rightmost point. The error after this first flood and proxy update is given by

$$E_1 = n \cdot \left\| \begin{pmatrix}0\\1\end{pmatrix} - N \right\|^2 + \left\| \begin{pmatrix}-1\\0\end{pmatrix} - N \right\|^2 .$$

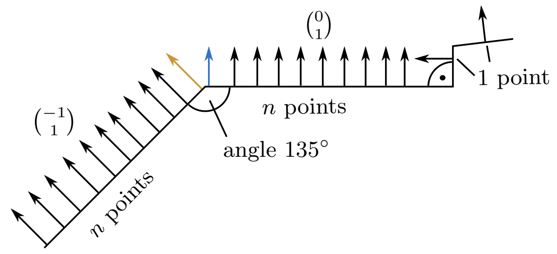
Starting from the new seed points, a second flood results in the situation shown in Figure 1(c). Here, almost all points except for the rightmost one are associated to the yellow proxy with normal $\binom{-1}{1}$. Its new normal after a proxy update is

$$N' = \frac{1}{2n+1}\left(n \cdot \begin{pmatrix}-1\\1\end{pmatrix} + n \cdot \begin{pmatrix}0\\1\end{pmatrix} + \begin{pmatrix}-1\\0\end{pmatrix}\right),$$

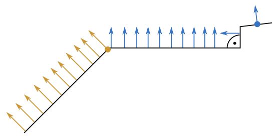
which amounts to an error after the second flood and proxy update given by

$$E_2 = n \cdot \left\| \begin{pmatrix}-1\\1\end{pmatrix} - N' \right\|^2 + n \cdot \left\| \begin{pmatrix}0\\1\end{pmatrix} - N' \right\|^2 + \left\| \begin{pmatrix}-1\\0\end{pmatrix} - N' \right\|^2 .$$

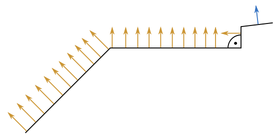
Choosing e.g. $n = 100$, we obtain $E_1 \approx 1.9802$, but $E_2 \approx 39.395$. Furthermore, the corresponding error value after the flood is also growing.



(a) Setup for growing error functional.



(b) Segmentation after first flood.



(c) Segmentation after second flood.

Figure 1: Example for a growth in the error measure after a flood and proxy update.

4 Modification for Converging VSA

In order to obtain an algorithm with guaranteed convergence, we propose to alter the steps of the algorithm as follows. First, we perform an initial *seeding*, one *flood* step, and a *proxy update* as explained above. Instead of the *seeding* step, we perform the following procedure:

4. *Switch*: For all points $p \in P$, consider their k nearest neighborhoods $\mathcal{N}_k(p)$. Assume that p is assigned to proxy P_i. If any point $p_\ell \in \mathcal{N}_k(p)$ is assigned to another proxy P_j, compute the change of the error measure in Equation (2.2) resulting from reassigning p from P_i to P_j. Compare it to the current best known reassignment. After iterating through all points $p \in P$, reassign the point such that the error measure is reduced maximally.

This new *switch* step replaces the *seed* step and the *flood* step described above. That is, it is only iterated together with the *proxy update*. The iteration is continued until no further *switch* operations can be performed. Although we describe the *switch* for point sets, it can easily be adopted for the mesh setting. For this alternate procedure, we can prove the following statement.

Theorem 1 (Error reduction by switch and proxy update, M. S. and E. Z.). *Given a point set $P = \{p_1, \dots, p_{n'}\}$ with a neighborhood structure, such that the neighborhood graph on P is connected and normals $n_1, \dots, n_{n'}$ on P, with n' denoting the number of points in P, then each proxy update step and each switch step as defined above leads to proxies (R_i, P_i) with a smaller error measure in Equation (2.2).*

Proof. Concerning the proxy update step, consider

$$\begin{aligned}\nabla E(\{R_i, P_i\}) &= \nabla \sum_{i=1}^{m} \mathcal{L}^{2,1}(R_i, P_i)\\ &= \sum_{i=1}^{m} \sum_{p_j \in R_i} \nabla \omega_j \left\| n_j - N_i \right\|_2^2\\ &= \sum_{i=1}^{m} \sum_{p_j \in R_i} 2\omega_j (n_j - N_i).\end{aligned}$$

Setting $N_i = \frac{\sum_{p_\ell \in R_i} \omega_\ell n_\ell}{\sum_{p_\ell \in R_i} \omega_\ell}$, that is updating the proxy normal as weighted average of its assigned normals, we obtain

$$\begin{aligned}\sum_{p_j \in R_i} 2\omega_j (n_j - N_i) &= \sum_{p_j \in R_i} 2\omega_j n_j - \sum_{p_j \in R_i} 2\omega_j \left(\frac{\sum_{p_\ell \in R_i} \omega_\ell n_\ell}{\sum_{p_\ell \in R_i} \omega_\ell} \right) \\ &= \sum_{p_j \in R_i} 2\omega_j n_j - \left(\frac{\sum_{p_\ell \in R_i} 2\omega_\ell n_\ell}{\sum_{p_\ell \in R_i} \omega_\ell} \right) \cdot \sum_{p_j \in R_i} \omega_j \\ &= \sum_{p_j \in R_i} 2\omega_j n_j - \sum_{p_\ell \in R_i} 2\omega_\ell n_\ell = 0.\end{aligned}$$

Thus, at the chosen updated proxy normal, the energy reaches a (local) minimum. As the energy is convex as sum of norms, which are convex, the found minimum is indeed its global minimum for the current choice of segmentation.

Concerning the *switch* step, only those points are reassigned which reduce the value of error measure (2.2). Therefore, trivially, after a *switch* operation the error is smaller. □

This theorem proves the convergence of our modified VSA procedure.

5 New Operations: Split and Merge

Two drawbacks of region growing approaches are the prescribed number of proxies to be chosen and the proper placement of seed points. The latter is often done manually in order to enhance results. In this section, we want to propose two new operations, which also adds adaptability of the algorithm to input and desired outcome. Also, they give the user the possibility to control the level of detail, i.e. how fine the segmentation should be in the end. For this, we introduce a user-given parameter $\kappa \in \mathbb{R}_{\geq 0}$ which controls the maximum deviation within a proxy region R_i from a corresponding completely flat approximation. This parameter is used in the following two additional steps:

(a) *Split*: Given a proxy P_i with its region R_i such that $\mathcal{L}^{2,1}(R_i, P_i) > \kappa$. We use weighted principal component analysis [2] to compute the most spread direction of R_i. The set R_i is then split at the center of this direction into two new regions $R_i = R_i^1 \sqcup R_i^2$. The new normals are chosen as $N_i^1 = \sum_{p_j \in R_i^1} \frac{\omega_j n_j}{\sum_{p_j \in R_i^1} \omega_j}$ and a corresponding N_i^2 respectively. The new centers C_i^1 and C_i^2 are then placed at those points of R_i^1, R_i^2 that have least varying normals from N_i^1 and N_i^2 respectively.
Note that the reasoning of Theorem 1 holds for this case, too. Thus, the modified VSA procedure outlined above, enriched with an additional split step does continue to converge.

(b) *Merge*: Consider a pair P_i, P_j of neighboring proxies with their respective normals N_i, N_j. If the region $R' = R_i \sqcup R_j$ with normal $N' = \frac{N_i + N_j}{2}$ achieves an error measure (2.2) strictly less than κ, the two regions are replaced by their union R', with normal N' and a chosen center $C' \in R'$ with its normal least deviating from N'.
Note that we could allow only those pairs of neighboring regions to merge such that

$$\mathcal{L}^{2,1}(R_i, P_i) + \mathcal{L}^{2,1}(R_j, P_j) \leq \mathcal{L}^{2,1}(R', P').$$

Then, the error measure would not increase and termination of the algorithm would be guaranteed. However, this would result in neighboring regions not observing the user-given κ threshold. Therefore, we accept an increase of the global error measure in favor of a better region layout. In all experiments performed, the algorithm still converged.

Both operations alter the number m of proxies. Thereby, a significant disadvantage of the algorithm is eliminated as the user does not have to choose m a priori. It is replaced by the user's choice of κ, providing a semantic guarantee on the regions being built by the algorithm. The user can prescribe a value of κ computed from the desired curvature within a proxy.

In the *merge* process outlined above, we asked for two neighboring regions. However, we have not defined any relation on the regions yet. In the meshed case discussed above, two regions are neighbors if and only if they share an edge in the mesh. In the context of point sets, we cannot rely on this, thus we propose the following definition. During the *flood* step described above, an element p is popped from the priority queue $\mathcal{Q}$ together with a possible assignment to a region R_i. However, it is ignored if p has already been assigned to a region R_j. In this case, we denote R_i and R_j to be neighbors, if $i \neq j$. This can be extended to the *switch* simply by considering two proxies to be neighbors, if in the k nearest neighborhood of a point $p \in P_i$, there exists a point $q \in P_j$, $i \neq j$.

This finishes the whole VSA pipeline, including the additional two steps *merge* and *split*. In the following, we show several results of the altered VSA on point sets.

6 Experimental Results

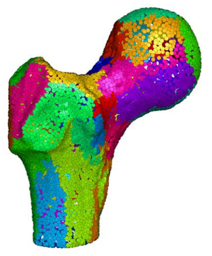
As our proposed extension of the VSA procedure gives some possibilities to arrange the steps, for the following experiments we used the pipeline:

Iterate(*seeding - flood - proxy update - one split - one merge*).

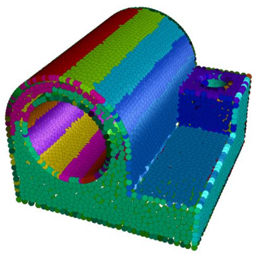
The displayed models with respective number of points in brackets are the CAD models Joint (9,998), Rockerarm (10,000), and Fandisk (6,475) as well as the real-world models Balljoint (10,002), Max Planck Bust (10,112), and Bunny (4,899), shown in Figures 2 and 3. The used parameters are given in the following table:

Model	m	m'	k	κ
Joint	15	34	8	20
Rockerarm	15	45	11	50
Fandisk	15	27	8	20
Balljoint	20	44	10	50
MaxPlanckBust	20	41	8	20
Bunny	15	36	8	20

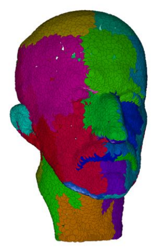
All these models visually give nice segmentation results, especially when we consider that all of them started with randomly selected seeds, where the number of seeds increased by the *merge* step in all cases to reflect the local geometries' behavior. A further visual comparison can be seen in Figure 3. The first row shows the segmentation gained with our approach, resulting in 27 (Fandisk) and 36 (Bunny) proxies. The second row shows results obtained by applying the VSA procedure with a corresponding number of seeds chosen randomly. Finally, there third row has been started with 15 triangle seeds for both models, which where selected manually for improved guidance of the algorithm, while additional seeds where added randomly to have the final seed numbers 27 (Fandisk) and 36 (Bunny). Here, we can see that our algorithm produces comparable results, despite the fact of not having manually placed seeds.



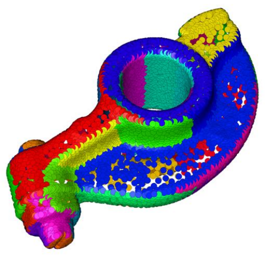
(a) Balljoint



(b) Joint



(c) Max Planck Bust



(d) Rockerarm

Figure 2: Our VSA adaption applied to four point-based models.

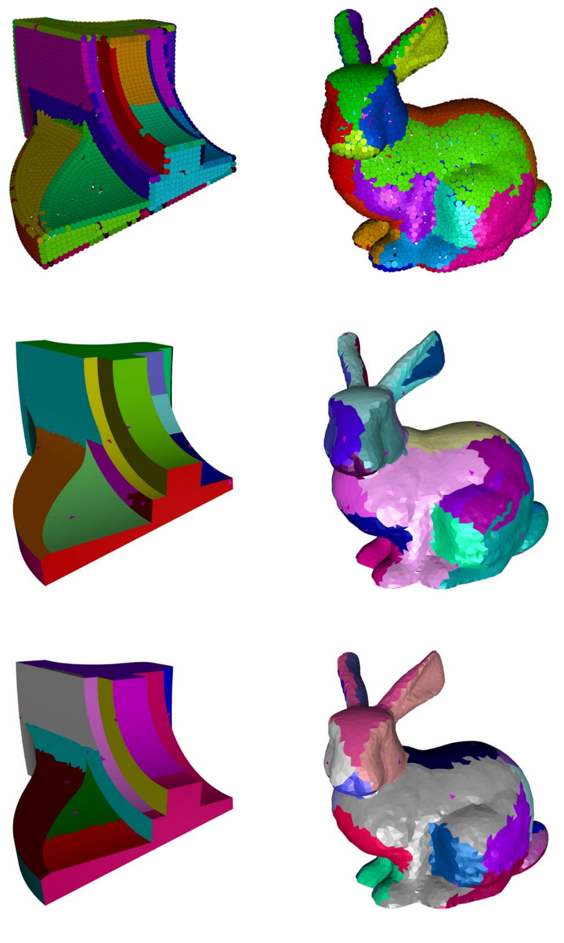


Figure 3: Segmentation applied to the Fandisk (left) and Bunny (right) models with our approach (first row), the VSA [1] with automatic (second row), and manual selected seeds (third row).

7 Conclusion

We have explained the VSA procedure, created an example with a growing error measure (2.1), proposed an alternate pipeline with the new operation *switch*, and gave proof of its convergence. Furthermore, we presented the two new operations *split* and *merge*, which make the procedure independent of a prescribed number of proxies and a (manual) initial selection of seeds as shown in our experiments.

For future work, we want to compare the quality of our segmentation approach with state-of-the-art methods and investigate a novel simplification algorithm based on the model approximating proxies.

References

[1] David Cohen-Steiner, Pierre Alliez, and Mathieu Desbrun. Variational Shape Approximation. In *ACM Transactions on Graphics (TOG)*, 2004.

[2] Paul Harris, Chris Brunsdon, and Martin Charlton. Geographically weighted principal components analysis. *International Journal of Geographical Information Science*, 2011.

[3] Adrien Kaiser, Jose Alonso Ybanez Zepeda, and Tamy Boubekeur. A survey of simple geometric primitives detection methods for captured 3d data. In *Computer Graphics Forum*, volume 38, pages 167–196. Wiley Online Library, 2019.

[4] Kai Wah Lee and Pengbo Bo. Feature curve extraction from point clouds via developable strip intersection. *Journal of Computational Design and Engineering*, 2016.

[5] Stuart P. Lloyd. Least squares quantization in pcm. *IEEE transactions on information theory*, 28(2):129–137, 1982.

[6] Martin Skrodzki, Johanna Jansen, and Konrad Polthier. Directional density measure to intrinsically estimate and counteract non-uniformity in point clouds. *Computer Aided Geometric Design*, 64:73–89, 2018.

An Optimized Yarn-Level Geometric Model for FEA Simulation of Weft-Knitted Fabrics

Paras Wadekar, Eric Markowicz, Dani Liu,
Genevieve Dion, Antonios Kontsos, David Breen [1]

[1] *Drexel University, Philadelphia, PA USA*

Abstract

Knitted fabrics are widely used in clothing and advanced textile devices because of their unique and programmable mechanical properties. In order to better understand and control these properties it is necessary to investigate the influence of yarn level interactions and stitch structure on the macroscropic behavior of the fabric via computational simulation, e.g. Finite Element Analysis (FEA). In this paper, we present a yarn-level model that produces geometric models suitable for FEA simulations of knitted fabrics. The geometric models of the yarns are produced via an optimization process. The centerlines of the yarns are defined as Catmull-Rom splines and their control points are modified during the optimization. The optimization is based on physical parameters such as interpenetration, contact, length of the yarn and bending energy. The results show that our approach produces valid yarn geometric models of knitted fabrics consisting of an arbitrary combination of knit and purl stitches, which can then be used for FEA simulations.

1 Introduction

The modelling and simulation of textiles has gained increased interest in recent years. These simulation efforts include the modelling and analysis of both woven and knitted materials. Our research focuses on advancing knitted textiles as a substrate for next-generation smart fabrics. A major thrust of this research is the development of design tools that will automate the specification of optimized knitted structures. A critical component of these tools are modelling and simulation technologies that are capable of accurately predicting the properties of a knitted material, given the properties of its yarns and the stitch patterns/commands used to knit the yarns into a fabric. To attain these goals simulations of knitted materials are done at the yarn-level using Finite Element Analysis (FEA) [5, 6].

A critical component of these simulations are the geometric models that define the initial configuration of the fabrics. In order for the simulations to properly proceed these geometric models must meet stringent requirements. The most important feature of the models is how they define contact between crossing yarns. Crossing yarns must "touch" at two points, but must not inter-penetrate each other. These contact points are defined by yarns that are outside of each other, but are within an extremely short distance to each other. A secondary requirement is that the initial geometric models by "plausible", i.e. they should not have unnatural, sharp bends or self-intersections.

A number of geometric models have been developed for knitted fabric simulation [2, 3, 4]. Other simulation models have been based on the topology of the fabric [1, 7]. None of these models though meet the strict contact requirements demanded by FEA in a general, parameterized approach.

In order to produce valid initial geometric conditions for FEA simulation studies we have implemented an enhanced yarn-level model of knitted fabrics that incorporates mechanical properties and spatial constraints with the underlying geometric representation of the yarns; thus producing initial parameterized geometric models that not only do not interpenetrate, but touch each other at point contacts, and additionally have a feasible, physically-accurate overall shapes. Our techniques produce yarn-level geometric models of weft-knitted fabrics consisting of an arbitrary pattern of knit and purl stitches. The dimensions of these individual stitches may be set by the user. Together these features allow for the generation of a wide variety of yarn-level geometric models that support the investigation of the relationship between yarn-level structures and macroscopic mechanical properties.

Producing physically-accurate geometric models of yarns in a knitted material is framed as an optimization problem. In this computing context, a single "cost" function is defined that captures the various required features of the final geometric model. The function is specified in such a way that finding the variable values that minimize the function produces the desired geometric result [8]. The features incorporated into the model, and therefore the associated cost function, include maintaining yarn length, minimizing curvature and creating single contact points between crossing yarns. The variables that are modified to minimize the cost function are the spline control points that define the centerlines of the tubes used to represent the yarns.

Once the optimization problem is formulated for a particular set of fabric parameters the cost function is minimized using a quasi-Newton method, which produces a spline that meets the requirements and constraints of the specified FEA initial conditions. This model has been utilized as the inputs to numerous FEA simulations of knitted materials. A number of output examples from the optimization process for a range of material size parameters are provided, which demonstrate the effectiveness of our approach to produce geometric models that are suitable initial conditions for FEA simulations.

2 Optimized Model

Since a fabric consists of repeated stitches our approach focuses on defining and optimizing the geometry of individual stitches, rather than the more computationally costly strategy of laying out the whole fabric and doing a global optimization. Once optimized, specific stitches, which we call cells, are replicated to produce the entire fabric. In order to maintain continuity between replicated cells, precise boundary conditions, both positional and tangential, must be defined and maintained for each cell.

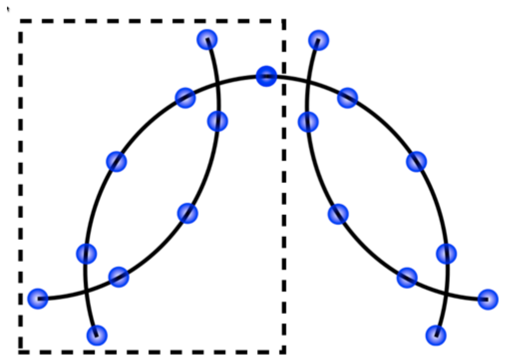


Figure 1: Control points and splines that define a single stitch.

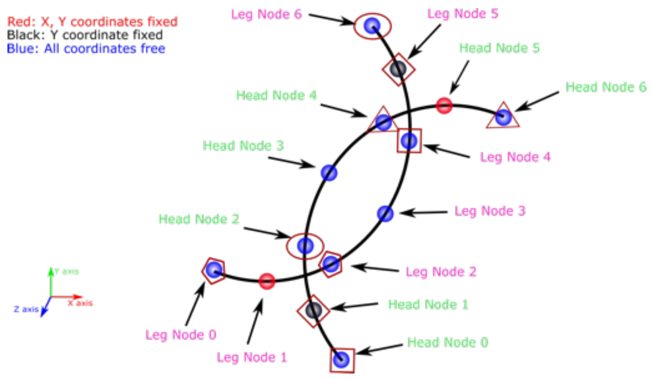


Figure 2: Constraints on the control points of unit cells.

2.1 Unit Cell for Optimization

Fig. 1 shows the control points and resulting splines that define the yarn geometry for a single stitch. The large single spline defines the "head" of stitch *i*, while the two smaller splines define the "legs" of stitch *i+1*, the one above it. The control points in the dotted box are considered a single cell, which can be reflected to produce the complete stitch. This stitch may then be copied and translated to produce a fabric model.

The following equations provide the parameterization of the stitch's dimensions in terms of course spacing (C), wale spacing (W) and yarn radius (R). These are used as initial positions of the control points to define the yarn in the left leg and the left half of the head of a single stitch.

$$\begin{aligned} P_{ix} &= a_{ix} * W \\ P_{iy} &= a_{iy} * C \\ P_{iz} &= a_{iz} * R \end{aligned} \tag{2.1}$$

The coefficients $\{a_{i\cdot}\}$ are obtained by defining an approximate yarn path with reasonable positions. The control points for the left half of the upper loop (head) of the stitch are assigned independently from those of the leg.

The yarn splines of knit (K) and purl (P) stitches are defined in a way that they are dependent on both neighboring stitches in the vertical direction, and the left stitch in the horizontal direction. Given this, there are only eight possible combinations of stitches that we need to consider. These eight combinations are: PPP, PPK, KPK, KPP, where four of them have the same neighbor as the center stitch while the other four have a different neighbor than the center stitch. Since a knit stitch is simply a reflection around the X-Y plane of a purl stitch, it is not necessary to consider separate cells with K as the center stitch. Thus eight distinct unit cells are able to create a stitch pattern of any size consisting of knits and purls.

While the curves of the unit cell are presented in Fig. 2, some of the control points needed to define these curves lie outside of the unit cell. This is because the tangent vector of the curve at a control point is defined by its neighboring control points. These extra, external control points needed to define the curves of the unit cell are Leg node 0, Leg node 6, Head node 0 and Head node 6. In order to maintain the correct tangent vector at the boundary control points (Leg node 1, Leg node 5, Head node 1, Head node 5) the external control points are linked with the control points interior to the boundary control points. For example, as Leg node 2 is moved during the optimization process, Leg node 0's position is updated in a way that it is a reflection (in the X component of Leg node 1) of Leg node 2. This ensures that the tangent direction at Leg node 1 remains parallel to the X-Z plane. The same holds true for Head node 4 and Head node 6. Head node 6's position follows Head node 4's in order to maintain the correct tangent direction ([1,0,0]) at Head node 5.

Similarly the correct spline shape, i.e. tangent, must be maintained across the unit cell boundaries at Head node 1 and Leg node 5. Note that during the copying and pasting process that creates a complete stitch these two control points define the same location. In other words the leg spline that exits the unit cell at Leg node 5 enters the head spline at Head node 1. Therefore the tangent vector at these two control points must be identical. This boundary constraint is enforced by having the position of Leg node 6 track/follow the position of Head node 2, and having the position of Head node 0 track the position of Leg node 4. All of the linking relationships are visually presented on Fig. 2, where unique symbols are placed around the linked control points. For example Leg node 0 and Leg node 2 are linked and both are marked with red pentagons. In all of these pairings, the interior control points are moved by the optimization process and the external control points' positions are then be updated.

2.2 Yarn Model Cost Function

A cost function $F(\cdot)$ is defined, based on the shape of the yarn splines. The minimum of this function represents a valid, physically-realistic geometric initial condition for an FEA simulation. $F()$ consists of three terms:

$$F(C^i(P^j)) = \alpha F_{Distance} + \beta F_{Bending} + \gamma F_{Length} \tag{2.2}$$

$F()$ is a function of the curves C^i that define the centerlines of the spline tubes used to represent the yarns. The centerlines are specified with Catmull-Rom splines, which consist of C1 piecewise cubic Bezier curves. The curves are defined by a set of control points P^j, which are interpolated by the curves C^i. $F_{Distance}$ is defined in such a way that it is at a minimum when the distance between the spline tubes is zero, i.e. the yarn geometries are not intersecting, and are touching at two points. $F_{Bending}$ defines the bending energy (as a function of curvature) of the yarn and ensures that the yarn geometry maintains a natural-looking shape. F_{Length} is the term that ensures that the yarn spline tube does not significantly change its length during optimization. α, β and γ are weights that control the strength of each term and give us the ability to adjust the importance/contribution of each component to the final result.

2.3 Optimizing the Unit Cells

Given a unit cell that consists of a leg spline and a head spline, an optimization may be employed to adjust the positions of the splines' control points in order to produce yarn geometric models with the desired properties. For design optimization purposes, we explore the parameter space of materials, e.g. course and wale spacing, as well as yarn thickness, in search of materials with optimal performance properties. Therefore it is important during FEA simulations of knitted materials that the simulated fabrics maintain certain size parameters. For this reason, not all of the spline control points are allowed to move freely.

Instead certain components of the yarn splines are constrained in order to ensure that the user-specified course and wale spacing is preserved after the optimization process has completed. The positions of the control points on the boundary of the unit cell (Leg node 1, Leg node 5, Head node 1, Head node 5 in Fig. 2) are therefore partially locked. The red nodes may only move in the Z direction during optimization, i.e. they are locked in the X and Y direction. The black nodes may move in the X and Z direction, i.e. they are locked in the Y direction. The positions of all the interior blue nodes are unconstrained during optimization. Again, constraining the boundary control points during optimization ensures that the values of C and W are maintained.

An unconstrained quasi-Newton method is utilized to minimize Equation 2.2 for the eight defined unit cells. This unconstrained optimization is performed while still enforcing the required constraints by simply omitting the fixed components of the constrained control points from the optimization process. Therefore 22 variables are involved in the optimization, i.e. the values of these variables are modified in order to minimize the cost function. These 22 variables are the X, Y and Z components of Leg node 2, Leg node 3, Leg node 4, Head node 2, Head node 3, and Head node 4 (the interior blue nodes in Fig. 2), just the Y component of Leg node 1 and Head node 5 (the red nodes), and the X and Z components of Leg node 5. Note that Head node 1 is not included in the optimization, since it is linked with Leg node 5 and tracks its position. Similarly the exterior control points (Leg node 0, Leg node 6, Head node 0 and Head node 6) are not directly modified by the optimization process, but instead have their positions determined by their linked, interior counterparts.

2.4 Replicating the Unit Cells

From a given stitch pattern the model is created by translating, reflecting and connecting the appropriate unit cells. The unit cells are designed to be the left halves of purl stitches. To produce the right half, a unit cell is simply mirrored across the X-Z plane. Similarly, to generate a knit stitch, a unit cell is mirrored across the X-Y plan.

The types (Knit or Purl) of the neighboring stitches determine the type of cell that will be placed, via replication and translation, at each stitch location in the fabric. Each cell has constraints imposed on it during optimization to ensure that C1 continuity is maintained across the boundaries to neighboring cells. This allows for seamless connections at the cell interfaces. Therefore any stitch pattern consisting of knit and purl stitches can be constructed by using these eight cells, and the resulting model may be used for an FEA simulation.

2.5 Writing an IGES file

Once the optimized control points for the entire fabric are generated, they are written to a file such that the geometry can be directly read by an FEA simulation software, such as ABAQUS. For this file we use the Initial Graphics Exchange Specification (IGES) format, as it can be read by a wide range of simulation softwares.

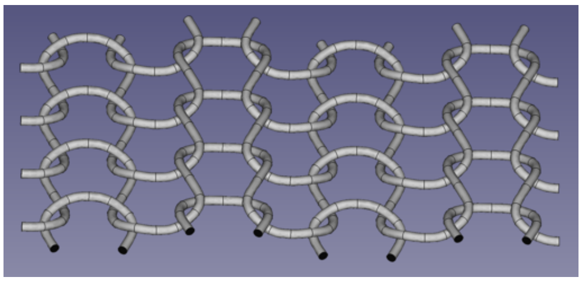


Figure 3: Rib pattern of 4x4 stitches.

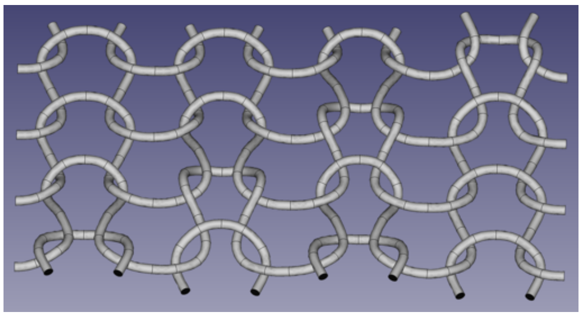


Figure 4: Random stitch pattern containing all the unit cells.

The control points of the individual cells are concatenated along each row to produce a single Catmull-Rom (CR) spline that defines the yarn along the row. The cross section of the yarn is assumed to be circular. A circle of the given yarn radius is swept through these CR splines to create a cylindrical tube. The ends of the yarns are capped by circular disks in order to create closed, solid objects. The IGES format requires the curves to be represented as Non-Uniform Rational B-Splines (NURBS). Hence the individual cubic Bezier curves of the CR splines are converted to NURBS of degree 3 while maintaining the same set of control points.

3 Results

It takes approximately 3 minutes for all unit cells to be generated and run through the MATLAB *fminunc* function on an Apple iMac with a 4.2 GHz Intel Core i7 processor and 32 GB of RAM.

For generating the examples, we have chosen the model's initial parameter values as W = 21.2, C = 12.8, R = 0.75. Using these values, different stitch patterns are generated. Fig. 8 shows a plain knit stitch pattern and Fig. 3 presents a rib pattern. It can be seen that only one unit cell is included in these patterns. Hence, a random stitch pattern of size 4×4 containing all eight unit cells was generated as shown in Fig. 4. It is defined as "pppk;ppkp;pkpp;kpkp", where ';' separates the rows. We use this pattern to compare models with different parameter values.

The model in Fig. 5 has an increased yarn radius. Fig. 6 increases course spacing by 25%, while in Fig. 7 the wale spacing is reduced by 25%. Fig. 8 presents the output of an FEA

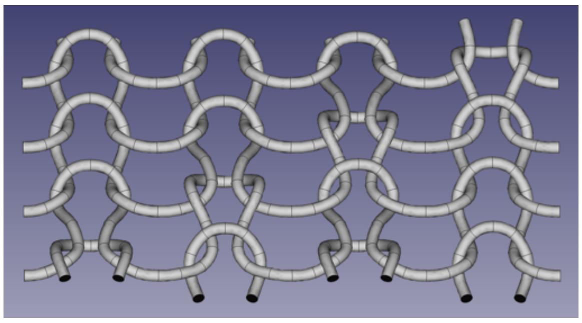


Figure 5: Random stitch pattern with yarn radius increased by 25%.

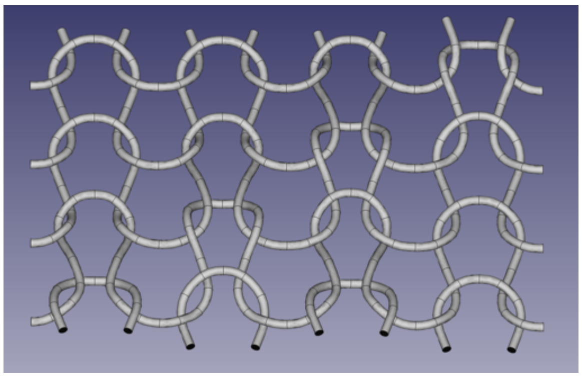


Figure 6: Random stitch pattern with course spacing increased by 25%.

simulation using our yarn-level geometric model.

4 Conclusion

We have presented a yarn-level geometric model of knitted fabrics that can be used for FEA simulations. The model is optimized to prevent inter-penetrations while minimizing the curvature and maintaining the length of the yarns. The control points of the model are used to create cylindrical surfaces representing the surface of the yarns. The model is written into an IGES file, which can then read by an FEA software such as ABAQUS for further processing.

This research was partially supported by National Science Foundation grant # CMMI-1537720.

References

[1] Gabriel Cirio, Jorge Lopez-Moreno, and Miguel Otaduy. Yarn-level cloth simulation with sliding persistent contacts. *IEEE Transactions on Visualization and Computer Graphics*, 23(2):1152–1162, February 2017.

[2] Jonathan M. Kaldor, Doug L. James, and Steve Marschner. Simulating knitted cloth at the yarn level. *ACM Trans. Graph.*, 27(3):65:1–65:9, August 2008.

[3] Y. Kyosev, Y. Angelova, and R. Kovar. 3D modeling of plain weft knitted structures of compressible yarns. *Research Journal of Textile and Apparel*, 9(1):88–97, 2005.

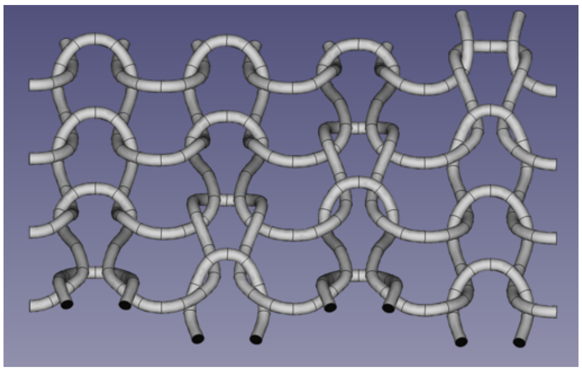


Figure 7: Random stitch pattern with wale spacing reduced by 25%

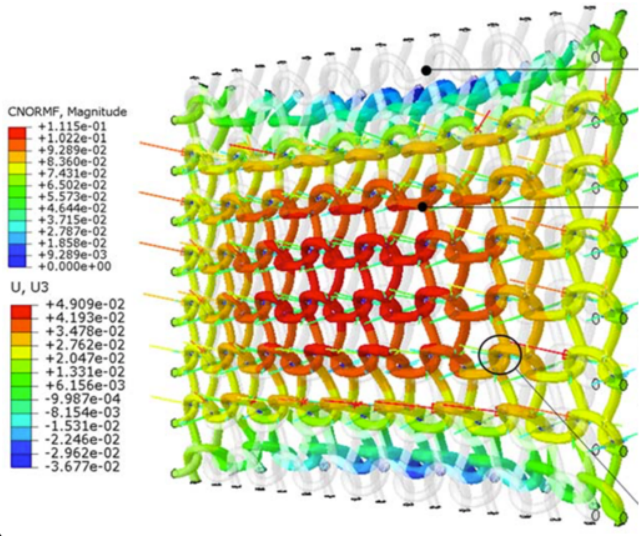


Figure 8: Results from a knitted fabric FEA simulation. Originally published in [6].

[4] H. Lin, X. Zeng, M. Sherburn, A.C. Long, and M.J. Clifford. Automated geometric modelling of textile structures. *Textile Research Journal*, 82(16):1689–1702, 2012.

[5] D. Liu, D. Christie, B. Shakibajahromi, C. Knittel, N. Castaneda, D. Breen, G. Dion, and A. Kontsos. On the role of material architecture in the mechanical behavior of knitted textiles. *International Journal of Solids and Structures*, 109:101–111, March 2017.

[6] D. Liu, B. Shakibajahromi, G. Dion, D. Breen, and A. Kontsos. A computational approach to model interfacial effects on the mechanical behavior of knitted textiles. *Journal of Applied Mechanics*, 85(4):JAM–17–1584, February 2018.

[7] M. Meissner and B. Eberhardt. The art of knitted fabrics, realistic & physically based modelling of knitted patterns. *Computer Graphics Forum*, 17(3):355–362, 1998.

[8] A. Witkin, K. Fleischer, and A. Barr. Energy constraints on parameterized models. In *Proc. ACM SIGGRAPH*, pages 225–232, July 1987.

Positive Geometries for Barycentric Interpolation

Márton Vaitkus [1]

[1] *Budapest University of Technology and Economics*

Abstract

We propose a novel theoretical framework for barycentric interpolation, using concepts recently developed in mathematical physics. Generalized barycentric coordinates are defined similarly to Shepard's method, using positive geometries – subsets which possess a rational function naturally associated to their boundaries. Positive geometries generalize certain properties of simplices and convex polytopes to a large variety of geometric objects. Our framework unifies several previous constructions, including the definition of Wachspress coordinates over polytopes in terms of adjoints and dual polytopes. We also discuss potential applications to interpolation in 3D line space, mean-value coordinates and splines.

1 Overview

The interpolation of scalar- or vector-valued data is an important task in many fields, including numerical analysis, geometric modeling and computer graphics. *Barycentric coordinates* can be defined over segments, triangles and simplices as well as more complex shapes, such as polytopes [4]. These include *Wachspress coordinates* [17] which are rational functions defined for convex polytopes. Wachspress coordinates were also generalized to subsets bounded by non-linear hypersurfaces [2] and can be defined in several equivalent ways [18, 11, 9]. We describe a theoretical framework that clarifies the relationship between these different formulations, and provides opportunities for novel generalizations.

We propose a set of basic building blocks for barycentric interpolation methods: *positive geometries*. Positive geometries have been recently introduced in the theoretical physics literature [1], but their application to interpolation problems has not been explored before. Besides simplices and polytopes, the category of positive geometries includes objects with similar combinatorial properties, such as polycons [17] or "positive" parts of toric varieties [15] and Grassmannians [12, 6]. A positive geometry carries a "canonical" differential form: a rational function with the properties of a signed volume, that has its poles (where the denominator vanishes) along the boundaries of the "positive" region. Crucially, these poles have a recursive structure, i.e. restricting a canonical form to a boundary component (via a generalization of taking complex residues) results in another canonical form. Thus, positive geometries share many properties with polytopes – most importantly, that complicated objects can be constructed by adding together simpler ones. To define interpolation in terms of canonical forms, we use a variant of Shepard's method [4, 13]: barycentric coordinates are ratios tending to $\frac{\infty}{\infty}$ along the interpolated boundaries. For polytopes, our construction recovers the definition of Wachspress coordinates in terms of dual volumes [9].

In this preliminary work, our goal is to introduce the theory of positive geometries to our audience, and demonstrate how they generalize earlier constructions for barycentric interpolation. After some motivating observations (Section 2), we give a definition of positive geometries and their canonical forms, also giving some examples (Section 3). We then describe how to use the canonical forms of positive geometries for barycentric interpolation (Section 4). Finally, we discuss potential applications of this framework to interpolation in line space, and possible extensions to non-rational barycentric coordinates and splines (Section 5).

2 Motivation

Let us consider the basic example of linear interpolation over a segment $[a, b] \subset \mathbb{R}$. The standard formula

$$f(x) = \frac{b-x}{b-a} f(a) + \frac{x-a}{b-a} f(b), \tag{2.1}$$

can be written in an alternative *barycentric form* [8]:

$$f(x) = \frac{\frac{1}{x-a} f(a) - \frac{1}{x-b} f(b)}{-\frac{b-a}{(x-a)(x-b)}}, \tag{2.2}$$

as a rational function with both the numerator and the denominator having poles at the boundaries a and b.

Consider next linear interpolation over triangles $(\mathbf{p}_0, \mathbf{p}_1, \mathbf{p}_2) \subset \mathbb{R}^2$ in the plane. The usual formula for barycentric interpolation gives

$$f(\mathbf{x}) = \frac{A_0 f(\mathbf{p}_0) + A_1 f(\mathbf{p}_1) + A_2 f(\mathbf{p}_2)}{A}, \tag{2.3}$$

which can be reorganized into the equivalent form

$$f(\mathbf{x}) = \frac{\frac{1}{A_1 A_2} f(\mathbf{p}_0) + \frac{1}{A_2 A_0} f(\mathbf{p}_1) + \frac{1}{A_0 A_1} f(\mathbf{p}_2)}{\frac{A}{A_0 A_1 A_2}}, \tag{2.4}$$

with the notations shown in Figure 1.

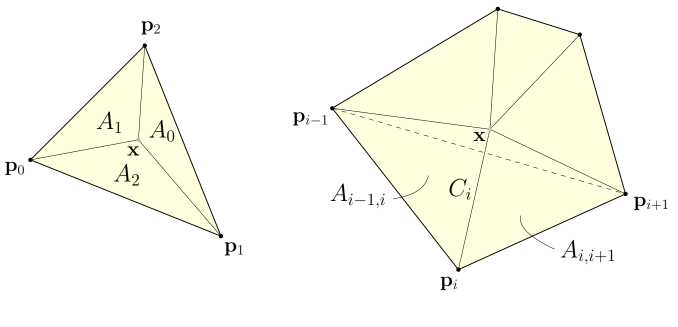


Figure 1: Notations for barycentric coordinates.

Wachspress proposed generalized barycentric coordinates over convex polytopes [17]. For a planar n-gon with vertices $\mathbf{p}_0, \dots \mathbf{p}_{n-1}$ these coordinates are defined as:

$$f(\mathbf{x}) = \sum_{i=0}^{n} \frac{\frac{C_i}{A_{i-1,i} A_{i,i+1}}}{\frac{\sum_k C_k A_{k-3,k-2} \cdots A_{k+1,k+2}}{\prod_k A_{k,k+1}}} f(\mathbf{p}_i), \tag{2.5}$$

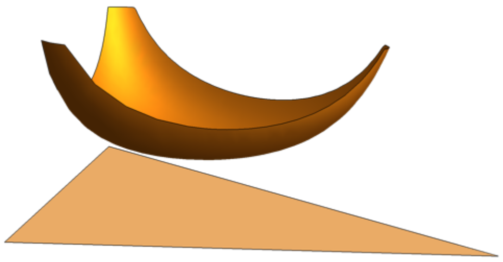


Figure 2: Canonical rational function of a triangle.

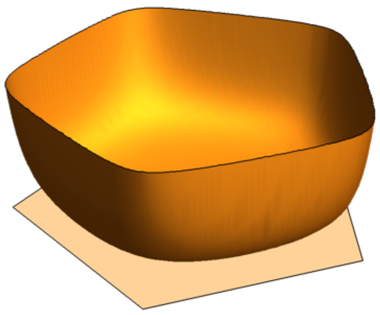


Figure 3: Canonical rational function of a pentagon.

where the indices are cyclical modulo n – see Figure 1.

In each of these cases, some terms in the numerator and denominator approach infinity along an interpolated subset, which is reminiscent of Shepard's method [8, 13]. The rational functions in the denominators are illustrated in Figure 2 and Figure 3. The common pattern involves functions with poles along the boundaries of some shape. Our goal is to make this idea rigorous using the notion of a positive geometry.

3 Positive Geometries and Canonical Differential Forms

The central concept of our work is that of a positive geometry – a relatively new concept originating from mathematical physics. In this chapter, we give an informal overview, and refer to [1] for technical details. For the necessary background in projective and algebraic geometry, see e.g. [12, 7].

3.1 Definition of Positive Geometries

Positive geometries were introduced as generalizations of shapes with a recursive structure similar to polytopes, defined by some sort of "positivity" criterion (e.g. the interior of polytopes, or totally positive matrices [5]). The definition was motivated by recent developments in quantum physics, where the solution of differential equations in Fourier space led to rational functions with poles at the boundaries of certain regions [1].

Definition 1. *A **positive geometry** is a pair $(X, X_{\geq 0})$, where X is a d-dimensional complex projective algebraic variety, and $X_{\geq 0}$ is an oriented semi-algebraic subset of its real part, with a unique differential d-form $\Omega(X, X_{\geq 0})$ called its **canonical form**, defined by recursion on the boundary dimension:*

- *If $d = 0$, then X is an (oriented) point, and $\Omega(X, X_{\geq 0}) = \pm 1$ depending on the orientation.*
- *If $d > 0$, then the boundary components of $X_{\geq 0}$ are themselves positive geometries, and the multivariate residue (see [1, A.3] or [7, Ch. 5]) along a component is the canonical form for the associated positive geometry.*

We stress that a positive geometry is determined by an ambient complex manifold (most often a projective space $\mathbb{P}^n$), together with a "positive" real subset, thus many different positive geometries could be associated to the same ambient space.

For positive geometries over projective spaces with homogeneous coordinates $\mathbf{x} = [x_0 : x_1 : \ldots x_d]$, the canonical form can always be written in terms of a rational function [1, C.1]:

$$\Omega(X, X_{\geq 0}) = C(\mathbf{x})\omega(\mathbf{x}), \tag{3.6}$$

where C is called the **canonical rational function** of the positive geometry and

$$\omega(\mathbf{x}) = \frac{1}{d!}x_0 dx_1 \wedge \ldots \wedge dx_d + \ldots + (-1)^d x_d dx_0 \wedge \ldots \wedge dx_{d-1}$$

is the *standard measure* on projective d-space.

3.2 Examples of Positive Geometries

Some elementary examples of positive geometries are the following:

- **Segments**, bounded by two points $\mathbf{a} = [a : 1]$ and $\mathbf{b} = [b : 1]$ in the projective line $\mathbb{P}^1$ parameterized using homogeneous coordinates. The canonical form at the point $\mathbf{x} = [x : y]$ is

$$\Omega(\mathbb{P}^1, [a, b]) = \frac{\langle \mathbf{b}, \mathbf{a} \rangle}{\langle \mathbf{x}, \mathbf{a} \rangle \langle \mathbf{x}, \mathbf{b} \rangle}\omega(\mathbf{x}), \tag{3.7}$$

  where $\langle \mathbf{v}, \mathbf{w} \rangle := \det(\mathbf{v}, \mathbf{w})$ denotes the determinant of vectors of homogeneous coordinates.

- **Simplices** in projective spaces. For triangles formed by three points $\mathbf{p}_i = [x_i : y_i : 1], i = 0, 1, 2$ in the projective plane $\Delta \subset \mathbb{P}^2$ parameterized using homogeneous coordinates $\mathbf{x} = [x : y : z]$, the canonical form is

$$\Omega(\mathbb{P}^2, \Delta) = \frac{1}{2}\frac{\langle \mathbf{p}_0, \mathbf{p}_1, \mathbf{p}_2 \rangle^2}{\langle \mathbf{x}, \mathbf{p}_0, \mathbf{p}_1 \rangle \langle \mathbf{x}, \mathbf{p}_1, \mathbf{p}_2 \rangle \langle \mathbf{x}, \mathbf{p}_2, \mathbf{p}_0 \rangle}\omega(\mathbf{x}). \tag{3.8}$$

Both of these forms are invariant under independent scaling of the homogeneous coordinates of the vertices, as well as the point of evaluation, and are thus well-defined functions over projective spaces.

Many other examples of positive geometries were identified [1, Ch. 5]:

- Planar regions bounded by a conic section and a line.
- Regions in projective 3-space bounded by a quadric or cubic surface and a plane.
- Positive, real parts of *Grassmannians* (manifolds of k-planes in n-dimensional space, denoted $G(k, n)$).
- Positive, real parts of *toric varieties* [15].

These examples – called **generalized simplices** – all have canonical forms with constant numerators.

The canonical forms of positive geometries are additive, i.e. the canonical form of a union is the sum of the canonical forms of its parts. The poles along boundaries meeting with opposite orientation will cancel, so that only poles on the exterior boundary remain (technically, this is true for a "signed triangulation of the empty set" – see [1, Ch. 3]). This implies that the canonical forms of more complicated regions can be determined by "triangulating" them. It follows that *convex polytopes* – which can be triangulated in the usual sense – are also positive geometries.

In analogy with generalized simplices, there are also various **generalized polytopes**:

- Convex regions of the projective plane bounded by straight lines and conics, which are examples of *polycons*, as defined by Wachspress [17].
- *Grassmann polytopes*, in particular *Amplituhedra*, which are generalizations of cyclic polytopes into Grassmannians.

Generalized polytopes have canonical forms with a non-constant numerator. In fact, for polytopes and polycons the numerator is known as the *adjoint polynomial* [10, 2].

3.3 Relation to dual polytopes

A canonical form often has a natural geometric interpretation. In particular, the canonical rational function of a convex projective polytope $P \subset \mathbb{P}^d$ is the signed volume of its *polar dual* $P^*_{\mathbf{x}} \subset (\mathbb{P}^d)^*$, as shown in [1, Ch. 7.4.1]. The polar dual is the intersection of the dual projective cone of the polytope, with the dual hyperplane of the point $\mathbf{x} \in P$, and its volume is a rational function over P computed by the integral formula

$$\mathrm{Vol}(P^*_{\mathbf{x}}) = \frac{1}{d!}\int_{\mathbf{y}\in P_{\mathbf{x}^*}} \frac{1}{(\mathbf{x}^T\mathbf{y})^{d+1}}\omega(\mathbf{y}). \tag{3.9}$$

4 Barycentric Interpolation over Positive Geometries

We claim that the canonical forms of positive geometries can be used to define barycentric interpolation in a way similar to Shepard's method.

Consider barycentric (linear) interpolation over a *segment.* Define the weight function for the endpoint a as the ratio of canonical forms for positive geometries over the projective line

$$\lambda_a(x) = \frac{\Omega(\mathbb{P}^1, [a, x^*])}{\Omega(\mathbb{P}^1, [a, b])}, \tag{4.10}$$

where x^* is the *projective dual* of the point x. If we choose coordinates so that x is the origin and x^* is the point at infinity, we get the barycentric formula (2.2) for linear interpolation.

The same construction applies to a *triangle* as well. For the the vertex $\mathbf{p}_i$, we take the ratio of two canonical forms over the projective plane – that of the original triangle, and the triangle bounded by the two sides meeting at $\mathbf{p}_i$ together with the dual line of the current point $\mathbf{x}$. Using the notations of Figure 4:

$$\lambda_i^{\Delta}(\mathbf{x}) = \frac{\Omega(\mathbb{P}^2, \Delta(l_{ki}, l_{ij}, \mathbf{x}^*))}{\Omega(\mathbb{P}^2, \Delta(l_{ij}, l_{jk}, l_{ki}))}. \tag{4.11}$$

This is simply the usual formula (2.4) for barycentric interpolation. Along each of the sides, the forms (technically, their residues) restrict to linear interpolation over a segment, as expected.

For a *convex polytope* $P \subset \mathbb{P}^2$ we follow the same recipe – the denominator for the vertex $\mathbf{p}_i$ will be the canonical form of P, while the numerator is the canonical form defined by the sides meeting at the vertex, with the dual line of the current point:

$$\lambda_i^P(\mathbf{x}) = \frac{\Omega(\mathbb{P}^2, \Delta(l_{(i-1)i}, l_{i(i+1)}, \mathbf{x}^*))}{\Omega(\mathbb{P}^2, P)}. \tag{4.12}$$

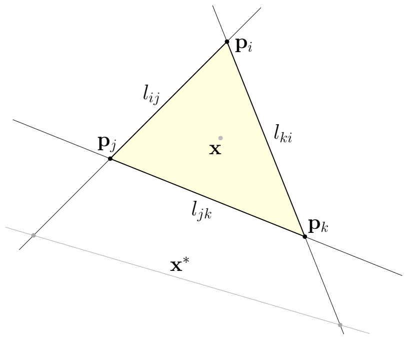


Figure 4: Triangles used for barycentric coordinates.

These weight functions are the Wachpress coordinates (2.5) over P. The generalization to higher-dimensional simplices and (simplicial) polytopes is straightforward.

In each case, the numerators are defined by positive geometries that form a signed triangulation of the domain. The additivity of canonical forms under unions then implies that their sum is the canonical form of the original polytope, and thus these functions form a *partition of unity.*

While these triangulations are not the most natural with respect to the original polytope, they correspond to a natural triangulation of the polar dual polytope including the origin (i.e. the current point $\mathbf{x}$). Recalling the interpretation of canonical forms as dual volumes, the numerator is seen as the volume of the dual pyramid formed by the current point and the dual face of the vertex. Thus, we connect to the earlier work of [9], who characterized Wachspress coordinates as ratios of polar dual volumes. Note that a triangulation of the polar dual through its vertices is analogous to Warren's triangulation-based definition of the adjoint polynomial for a polytope [18].

Wachspress coordinates can be generalized also to regions bounded by subsets bounded by higher-order algebraic varieties, such as *polycons* [17]. We omit the discussion of these cases to conform to spatial limitations.

5 Discussion and Future Work

Our approach to barycentric interpolation with canonical forms of positive geometries unifies earlier approaches using adjoint polynomials, dual volumes and Shepard-like interpolation. An advantage of this framework is that it extends to positive geometries other than polytopes, such as Grassmannians and toric varieties.

We also mention that the definition of a positive geometry embeds the domain into a higher-dimensional complex manifold, thus our approach can be viewed as a multivariate generalization of complex analytical methods (contour integrals, residues) used in univariate approximation theory [16].

5.1 Open Problem: Interpolation in Grassmannians

As was mentioned previously, certain "positive" subsets of Grassmannians are also examples of positive geometries. Points in a Grassmannian $G(k, n)$ can be represented by $k\times n$ matrices, and the positive geometry known as the *positive Grassmannian* corresponds to *totally positive* matrices, which are of great interest for approximation theory and geometric modeling [5].

The Grassmannian of 2-planes in 4-space, $G(2, 4)$ – which is also the manifold of lines within 3-space – is particularly interesting for many applications [12]. $G(2, 4)$ is a 4-dimensional hypersurface in $\mathbb{P}^5$ cut out by a quadratic equation in Plücker coordinates, and the positive Grassmannian is a semi-algebraic subset with non-linear boundaries. The line-geometric analogue of a simplex might be related to the *tetrahedral line complex*, the boundary of which is the set of lines defined by the edges of a tetrahedron. The combinatorial structure of this boundary is that of an octahedron – an example of a *hypersimplex* – as shown in Figure 5, where each vertex represents a line along an edge of the tetrahedron, while each face represents lines through either a vertex or a face [6].

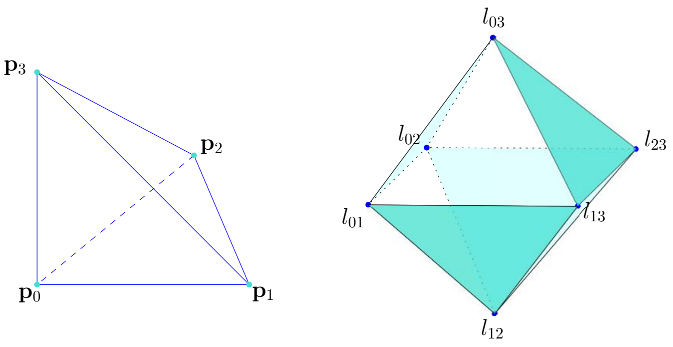


Figure 5: Tetrahedral line complex and corresponding hypersimplex. Shaded faces correspond to lines through vertices.

Being a positive geometry, the positive Grassmannian has a canonical form, i.e. a rational function of Plücker coordinates with poles along its boundaries [1, Ch. 5.5]. This suggests that generalizations of barycentric coordinates into line space might be possible using our framework.

5.2 Open Problem: Generalized Positive Geometries for Mean-Value Coordinates

Canonical forms are defined by rational functions, so Mean-Value Coordinates (MVCs) [4] – defined by transcendental functions – are apparently incompatible with the proposed framework. We could adapt the approach of [14], where MVCs are expressed as dual Shepard interpolants, after deforming the original boundary to a unit circle around the current point. A disc is not a positive geometry in the usual sense – it lacks zero-dimensional boundary components, for example. Nevertheless, we can easily find a projectively well-defined function with singularities along a projective conic $\mathcal{C}$ given by the quadratic equation $\mathbf{x}^T\mathbf{Q}\mathbf{x} = 0$ (see [1, Ch. 10] for a lengthy discussion):

$$\Omega(\mathbb{P}^2, \mathcal{C}) = \frac{\pi \det(\mathbf{Q})^{\frac{3}{4}}}{(\mathbf{x}^T\mathbf{Q}\mathbf{x})^{\frac{3}{2}}}\omega(\mathbf{x}). \tag{5.13}$$

Observe the similarity with the transfinite form of MVCs [4, Ch. 10], when $\mathcal{C}$ is a circle. This kind of canonical function is not rational and its singularities along $\mathcal{C}$ are not simple poles, but *branch points* (similar to the origin of the complex plane for fractional powers and logarithms). The authors of [1] also identified such transcendental generalizations of positive geometries as promising subjects for future research.

5.3 Open Problem: Relation to Splines and Integral Geometry

Formulas such as the dual volume (3.9) also appear in the context of multivariate (box/simplex/cone) splines [3], as Laplace transforms of indicator functions. This suggests that splines and barycentric interpolants could both fit within an even more general theoretical framework related to integral geometry.

Acknowledgements

This project has been supported by the Hungarian Scientific Research Fund (OTKA, No.124727). The author thanks Tamás Várady for support and Péter Salvi for interesting discussions.

References

[1] N. Arkani-Hamed, Y. Bai, and T. Lam. Positive geometries and canonical forms. *Journal of High Energy Physics*, 2017(11):39, 2017.

[2] G. Dasgupta and E. Wachspress. The adjoint for an algebraic finite element. *Computers & Mathematics with Applications*, 55(9):1988–1997, 2008.

[3] C. De Concini and C. Procesi. *Topics in Hyperplane Arrangements, Polytopes and Box-Splines.* Springer, 2010.

[4] M.S. Floater. Generalized barycentric coordinates and applications. *Acta Numerica*, 24:161–214, 2015.

[5] M. Gasca and C.A. Micchelli, editors. *Total Positivity and Its Applications.* Kluwer, 1996.

[6] I.M. Gelfand and R.D. MacPherson. Geometry in Grassmannians and a generalization of the dilogarithm. *Advances in mathematics*, 44(3):279–312, 1982.

[7] P. Griffiths and J. Harris. *Principles of Algebraic Geometry.* John Wiley & Sons, 1978.

[8] K. Hormann. Barycentric interpolation. In *Approximation Theory XIV: San Antonio 2013.* Springer, 2014.

[9] T. Ju, S. Schaefer, J. Warren, and M. Desbrun. A geometric construction of coordinates for convex polyhedra using polar duals. In *Proceedings of the 3rd Eurographics Symposium on Geometry Processing.* Eurographics, 2005.

[10] K. Kohn and K. Ranestad. Projective geometry of Wachspress coordinates. *arXiv preprint arXiv:1904.02123*, 2019.

[11] M. Meyer, A. Barr, H. Lee, and M. Desbrun. Generalized barycentric coordinates on irregular polygons. *Journal of graphics tools*, 7(1):13–22, 2002.

[12] H. Pottmann and J. Wallner. *Computational Line Geometry.* Mathematics and Visualization. Springer, 2001.

[13] V.L. Rvachev, T.I. Sheiko, V. Shapiro, and I. Tsukanov. Transfinite interpolation over implicitly defined sets. *Computer Aided Geometric Design*, 18(3):195–220, 2001.

[14] S. Schaefer, T. Ju, and J. Warren. A unified, integral construction for coordinates over closed curves. *Computer Aided Geometric Design*, 24(8-9):481–493, 2007.

[15] F. Sottile. Toric ideals, real toric varieties, and the algebraic moment map. In *Topics in Algebraic Geometry and Geometric Modeling.* AMS, 2003.

[16] L.N. Trefethen. *Approximation Theory and Approximation Practice.* SIAM, 2013.

[17] E. Wachspress. *Rational Bases and Generalized Barycentrics: Applications to Finite Elements and Graphics.* Springer, 2016.

[18] J. Warren. Barycentric coordinates for convex polytopes. *Advances in Computational Mathematics*, 6(1):97–108, 1996.

CHoCC: Convex Hull of Cospherical Circles

Yaohong Wu[1], Ashish Gupta[1], Kelsey Kurzeja[1], and Jarek Rossignac[1]

[1] *School of Interactive Computing, Georgia Institute of Technology, Atlanta, USA*

Abstract

We discuss the properties and computation of the exact boundary B of the ***convex hull***, H($\mathcal{C}$), of a set $\mathcal{C}$ of n oriented circles $\{C_i\}$, which may have different radii, but all lie on the same sphere, S, and bound disjoint caps. The faces of H comprise: the n ***caps***, $t = 2n - 4$ ***triangles***, each having vertices on different cap borders, and $3t/2$ developable surfaces, which we call ***corridors***. The connectivity of H and the vertices of its triangles may be obtained by computing the Apollonius diagram of a flattening of the caps via a stereographic projection. The corridors are each a subset of an elliptic cone and their four vertices are coplanar. The above ideas may be used to compute and process the ***Convex Hull of Cospherical Circles*** (***CHoCC***), in which each cap of H is replaced by the corresponding disk. A lattice may be approximated by an ***ACHoCC***, which is an ***Assembly*** of touching, but not interfering, CHoCCs that share ***contact disks***. The simplicity of the boundary representation of each CHoCC and of their disk-interfaces makes the ACHoCC attractive for lattice processing and 3D printing. We also discuss polyhedral coarse approximations of CHoCCs and ACHoCCs.

1 Introduction

In this paper, we propose a representation of the exact convex hull, H($\mathcal{C}$), of a set $\mathcal{C}$ of n oriented circles, $\{\mathrm{C}_i\}$, of possibly different radii, that all lie on the same sphere, S (Fig. 1-left). Throughout this paper, we assume that the spherical caps defined by a chosen orientation of each one of these circles are pairwise disjoint.

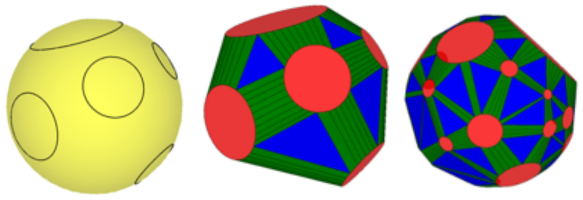


Figure 1: Circles around disjoint caps on a sphere (left) and their convex hull (middle), comprising disks (red), triangles (blue), and corridors (green), each bounded by two straight and two circular edges. We draw (black) a sampling of generators on each corridor. We show (right) an example with more circles.

In Sec. 2, we show that the face-graph of the convex hull of $n > 2$ cospherical circles has $6n - 10$ faces. Each face is either a disk, a triangle, or a developable surface, which we call a ***corridor*** (Fig. 1-middle). We also explain how it may be computed efficiently by constructing the Apollonius diagram [7].

In Sec. 3, we point out that each corridor is a subset of an elliptical cone. This result was known in Classical Geometry [10]. We also explain how to compute its axis and apex.

In Sec. 4, we discuss an application for approximating a lattice by an assembly of solid elements (one per junction and one per beam), each being the Convex Hull of Cospherical Circles (***CHoCC***). The interiors of the elements are disjoint. Each junction element touches incident beam-elements at disk-interfaces.

Finally, in Sec. 5, we discuss the generation of tessellations of these circles and their use for generating coarse polygonal-mesh approximations of any selected portion of a lattice.

2 Face-graph topology and vertices

In our representation of the boundary B of H($\mathcal{C}$) and in our algorithm, we exploit the following homeomorphism between the adjacency graph of these faces and the connectivity graph of a triangle mesh, $\mathcal{M}$: disks (resp. corridors, triangles) of B map to the vertices (resp. edges, triangles) of $\mathcal{M}$. The mapping becomes obvious if we shrink each disk to its center. Hence, the boundary B of the convex hull of n cospherical circles has the following face types and counts: (1) the n ***disks***, each being the convex hull of an input circle C_i, (2) the $t = 2n - 4$ ***triangles***, each connecting three of the disks, (3) the $3t/2$ developable patches, which we call ***corridors***, each adjacent to two disks and two triangles (Fig. 1-right). Consequently, the total face-count is $n + (2n - 4) + 3(2n - 4)/2$, which is $6n - 10$.

For each triplet $\{\mathrm{C}_i, \mathrm{C}_j, \mathrm{C}_k\}$ of circles in $\mathcal{C}$, there exists 2 supporting planes. Each such supporting plane, Π, touches each of these three circles at a vertex of a candidate triangle, $\mathrm{T}_{i,j,k}$. We orient $\mathrm{T}_{i,j,k}$ and Π such that the circles $\{\mathrm{C}_i, \mathrm{C}_j, \mathrm{C}_k\}$ lie inside the associated halfspace, Π^+. A candidate triangle is ***valid***, i.e., a triangle of B, if and only if, all the other circles of $\mathcal{C}$ are also in Π^+.

The intersection of S with Π is the circumcircle of $\mathrm{T}_{i,j,k}$. Hence, computing $\mathrm{T}_{i,j,k}$ is equivalent to finding the circle on S that is tangent to C_i, C_j, and C_k (there exists eight such circles, see Fig. 2-left) and that lies on a supporting plane, Π (there exists two such circles, see Fig. 2-right). This is a version of the Apollonius' problem, but on a sphere.

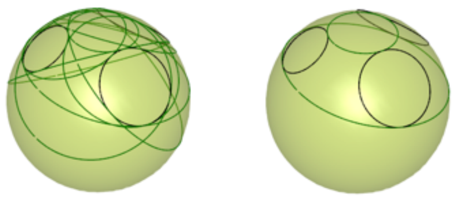


Figure 2: Eight circles (dark green) tangent to three given circles (black) on a sphere (left), and the two possibly valid solutions that each lies on a supporting plane of the three given circles (right).

Starting with the n spherical caps defined by the oriented input circles $\{\mathrm{C}_i\}$, the vertices of H($\mathcal{C}$), may be computed by the following sequence of steps:

1) Transform the spherical caps into a set of disks in the plane through a stereographic projection [3], $\mathbf{P}$, which preserves angles and circles.

2) Compute the Apollonius diagram of the planar arrangement of disks. This computation may be performed in CGAL [7]. The Apollonius diagram is also called the additively weighted Voronoi diagram. Its computation is discussed in [8]. The diagram defines maximal circles (see blue circles on the plane in Fig. 3) that are each tangent to three different disks at three contact-points. Each triplet of such points that is associated with a maximal circle corresponds to the three vertices of a valid triangle, $\mathrm{T}_{i,j,k}$, of $\mathrm{H}(\mathcal{C})$.

3) For each triplet, record the associated disk-indices, $\{i, j, k\}$, transform the contact-points back onto S using the inverse of $\mathbf{P}$ and produce the corresponding triangle (blue).

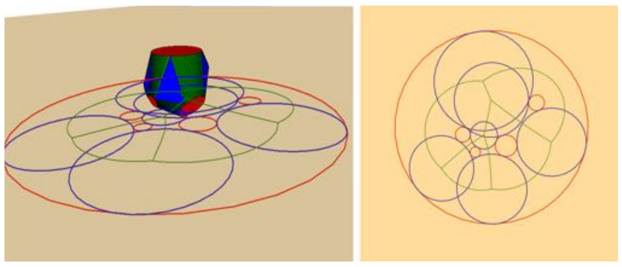


Figure 3: Left: 3D view of 5 cospherical circles (red) with 2D Apollonius diagram drawn on a plane parallel to the top red disk. Right: 2D view of the diagram (green), showing maximal-circles (blue), each tangent to three red circles.

From this information, we can easily reconstruct the blue triangles that each touch three red disks, and hence the border edges of each green corridor.

3 The surface of each corridor

The convex hull in three dimensions of two circles in general configuration has faces that are ruled surfaces of degree eight [9].

Remarkably, when the two circles lie on a sphere, their convex hull is bounded by faces that are either planar (the two disks in situations where the caps are disjoint) or subsets of elliptical cones. Indeed, each circle is the intersection of a plane with the sphere, S. Spain [10] proved that in the special case of intersection between two quadric surfaces, wherein one of the quadrics degenerate into a pair of planes, there exists two cones that pass through their intersection. Hence, there exists two elliptic cones that pass through two cospherical circles (Fig. 4).

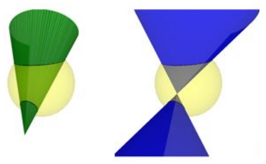


Figure 4: There exists two elliptic cones passing through two disjoint circles on a sphere S. One with its apex outside of S (left)—the other inside (right).

Our corridors are each a subset of an elliptical cone through two input circles, and the two straight edges bounding a corridor are generators of that cone. As shown in Fig. 5, a corridor, R, is adjacent to two red disks, D_i and D_j and two blue triangles $\mathrm{T}_{i,j,k}$ and $\mathrm{T}_{j,i,l}$. It is defined by the ordered quadruplet of circles $\{\mathrm{C}_i, \mathrm{C}_j, \mathrm{C}_k, \mathrm{C}_l\}$.

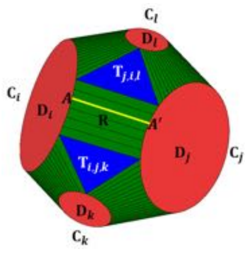


Figure 5: A corridor is defined by a quadruplet of circles.

The corridor is the union of a pencil of ***generators*** (line segments). To render a corridor as a quad mesh or to compute its silhouette, we use a parameterization of these generators. Given a point A (one end of a generator) on C_i (Fig. 5) and the apex F of the cone through C_i and C_j, we compute the corresponding point A' (the other end of that generator) on C_j as the intersection of plane containing C_j with the line through F and A.

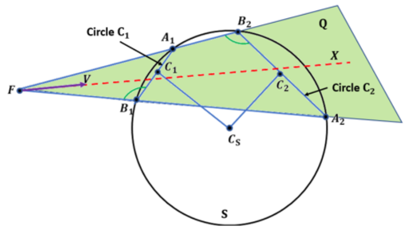


Figure 6: The apex F of the elliptic cone (green) is the intersection of line $\mathrm{L}(A_1, B_2)$ with line $\mathrm{L}(B_1, A_2)$. The red dotted line is the axis of the elliptic cone, it bisects $\angle AFB_1$.

To compute the apex F of an elliptical cone that contains two circles, C_1 and C_2, with centers C_1 and C_2 on sphere S with center C_s, we consider the plane π passing through C_1, C_2 and C_s. We consider the intersections A_1 and B_1 of C_1 with π and also the intersections A_2 and B_2 of C_2 with π, as shown in (Fig. 6). F is the intersection of the line through A_1 and B_2 with the line through A_2 and B_1.

4 Approximating a lattice by an ACHoCC

One motivation for this work is to simplify and accelerate the processing of extremely complex lattices represented by parameterized programs [5]. Such a lattice may contain billions of beams and it may not be possible to fully evaluate its boundary nor even the parameters of all its elements. Hence, we rely on lazy (on demand) evaluation of a selected subset of the model to support various queries for analysis or printing.

Gupta et al. [5] define a ***lattice*** in terms of a union of ***balls*** and cone ***beams***, which may be decomposed into an assembly of ***hubs***. A beam F is a solid conical frustum defined to smoothly connect two balls, B_1 and B_2, such that the union $\mathrm{F} \cup \mathrm{B}_1 \cup \mathrm{B}_2$ is the convex hull of B_1 and B_2. A ***half-beam*** is a portion of a beam, cut by the plane that is equidistant from B_1 and B_2 and is perpendicular to the axis through the centers of B_1 and B_2. A ***hub*** is the union of a ball and of every half-beam incident on it. A lattice is composed of hubs (Fig. 7), which we assume do not intersect one another, except at the disks capping each half-beam.

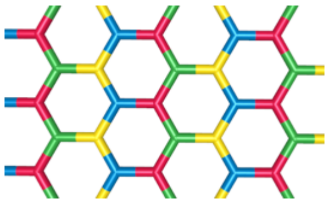


Figure 7: A portion of a lattice with individual hubs shown in separate colors.

Even though the geometry and topology of each hub is relatively simple [4], we discuss here an approach that simplifies that geometry and accelerates its evaluation.

The key idea is to associate a sphere S, with each hub. S has the same center as the ball of the hub. But its radius is slightly larger than the smallest radius for which the half-beams of the hub do not interfere outside of S.

Replacing each hub by its union with the solid bounded by S produces an ***inflated-hub*** approximation of the hub. It has a larger volume, but a simpler boundary: only circular edges.

We split the inflated-hubs at ***contact-planes*** that contain their circular edges and glue the remaining parts of the half-edges back together. We obtain a new decomposition of the lattice into an assembly of ***chopped-balls*** (intersection of the ball bounded by S with halfspaces bounded by the contact planes) and ***chopped-beams*** (conical frustums). Notice that these chopped-balls and the new beams are all convex and have only circular edges. (We are assuming that their interiors are all disjoint). A chopped-ball touches incident chopped-beams along ***contact-disks***.

To reduce the added volume, we replace each chopped-ball by the convex hull of its contact-disks. Observe that each element of this new model is a CHoCC (Fig. 8-left), and the whole lattice is an assembly of CHoCCs.

5 Tessellation

For some applications, we tessellate the boundaries of hubs into coarse approximations without evaluating non-selected portions of the lattice.

The natural, minimal tessellation of a corridor is into a single quad that interpolates the four coplanar vertices of the corridor (Fig. 8). Each disk is replaced by a polygon that interpolates the vertices on its boundary. So, that tessellated version of a CHoCC comprises its original triangles, a quad per corridor and a polygon per disk.

The case of a convex hull of only two circles must be handled specially, because it contains only one corridor, which is tessellated with a minimum of three planar quads.

The minimal tessellation of a hub may produce non-manifold solids or may result in unacceptable approximation errors (either with respect to the ACHoCC or to the corresponding parts of the original lattice) for the target application, so we discuss below post-processing tools for improving accuracy.

The first step in improving the quality of the tessellation is to refine it. We insert a vertex at the midpoint of each edge of the mesh of the CHoCC of each chopped-ball. For each triangle of that mesh, we refine it into four new triangles, where one of the new triangles is incident on the three inserted vertices. For

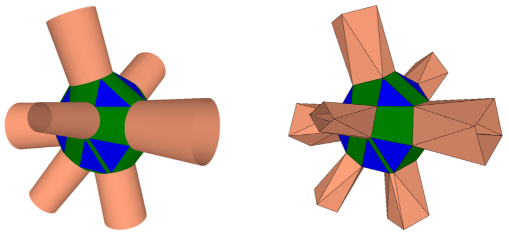


Figure 8: The surface of the convex hull of circles on a sphere unioned with the surfaces of the conical beams (left). Their coarsest (natural) approximation in which each corridor is represented by a single planar quad (right).

each quad, we insert one new vertex at its centroid and refine it into eight new triangles (Fig. 9-left).

The refine step does not change the shape of the mesh, and so does not improve its approximation accuracy. To improve accuracy, a vertex that was inserted on an edge of a polygonal approximation of a disk is snapped radially onto the circle bounding that disk (Fig. 9-right). Note that after this process, the mesh may no longer be convex.

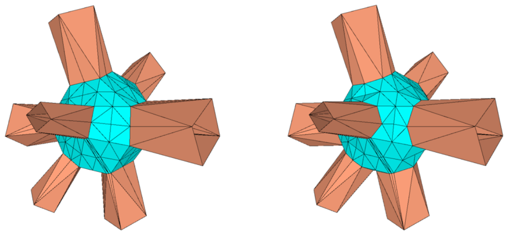


Figure 9: (Left) A subdivision of the mesh of Fig. 8-right. (Right) The result of snapping the polygonal vertices onto their circle.

The tessellation described above differs from the original lattice in two significant ways: 1) its volume may be increased when the CHoCC of a hub is much larger than the ball, 2) some of the hub ball may lie outside of the corresponding CHoCC.

We amend both of these problems with a central projection step, which projects each vertex onto the surface of the exact hub. If the vertex is outside of the exact hub boundary, then we move it to the first intersection of the ray from it to the hub's center. Otherwise, we move it to the first intersection of the ray from the hub's center to the vertex (Fig. 10-left). The central projection requires that the hub's center be contained by the CHoCC. We are experimenting with alternative projections without this requirement.

A fillet may be desired, in place of sharp edges at the junctions between beams, to reduce stress concentrations. The filleted hub is represented as an implicit surface, based on Blinn's blobby molecules [2]. The equation for a filleted hub is

$$d(P) = -\frac{\ln \sum_{i=1}^{n} \exp\left(-b d_i(P)\right)}{b} \tag{5.1}$$

where $d(P) = 0$ when point P is on the surface of the filleted hub, $d_i(P)$ is the signed distance from P to the i^{th} beam (unioned with the two balls it connects), and $b > 0$ is the blobbiness parameter where smaller values yield a larger fillet. We implement $d_i(P)$ as the cone-sphere distance computation from Barbier et al. [1], and we use sphere tracing to solve the ray intersection against the filleted hub [6] (Fig. 10-right).

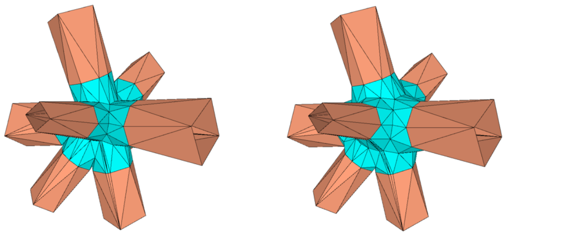


Figure 10: The inserted vertices are projected onto the surface of the hub by moving the vertices towards the center of the hub's sphere (left). A filleted version of the junction (right) obtained by projecting the vertices on onto a blending surface.

We can construct, on demand, any of the variants discussed above for a selected portion of the lattice. For example, figure 11 shows the crudest tessellation of the lattice without refinement.

Xiong et al. [13] proposed to compute structured quad meshing (following natural parameterization of underlying surface) of a network of tubular surfaces. Although their proposed formulation produces good quality quad meshes, it is globally constrained over all branches of the tubular network and may not be viable for meshing lattices with large number of beams. Stasuik and Piker [12] proposed a Grasshopper plugin called "Exoskeleton" to construct the triangle mesh of the surface of a solid obtained by thickening of a wireframe model [11]. Srinivasan et. al. [11] use convex hull of polygonal faces of beams connected to a joint to approximate joints in a network of cylindrical pipes and then cleanup the convex hull by removing redundant edges. Compared to that, our approach works with cone beams, computes exact convex hull of a set of circular end faces of beams, and directly produces the lowest poly approximation of the exact convex hull.

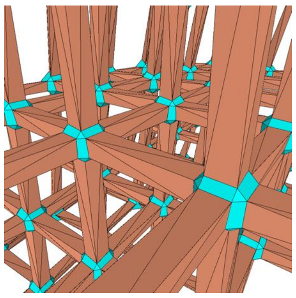


Figure 11: A coarse approximation (without refinement) of a lattice with convex polyhedra for chopped-beams (brown) and chopped-balls (cyan).

6 Conclusions

We show that the topology and geometry of the convex hull of $n > 2$ cospherical circles is surprisingly simple. It has $6n - 10$ faces that are each either a disk, a triangle, or a corridor that is a portion of an elliptic cone bounded by two line segments and two circular arcs. We suggest a simple and efficient approach for computing the connectivity of that convex hull, its vertices, and the apex of the cone of each corridor. We discuss a possible application for computing an assembly of simple solids that approximates a given lattice by the union of Convex-Hulls of Cospherical Circles (CHoCC). Each CHoCC corresponds to a beam or a junction of the lattice. The CHoCC of each junction touches the CHoCC of each of its incident beams along a disk. But the interiors of the CHoCCs are disjoint. We discuss tessellations of this model that offer different degrees of accuracy.

Acknowledgements

This research was developed with funding from the Defense Advanced Research Projects Agency (DARPA). The views, opinions and/or findings expressed are those of the author and should not be interpreted as representing the official views or policies of the Department of Defense or the U.S. Government.

We thank Alexander Pasko and his lab for providing the blending function used to generate the filleted version of the junctions. We also thank Helmut Pottmann for his comments on how results from Classical Geometry may be used to prove that the convex hull of two consperical circles is bounded by an elliptic cone.

References

[1] Aurelien Barbier and Eric Galin. Fast distance computation between a point and cylinders, cones, line-swept spheres and cone-spheres. *Journal of Graphics tools*, 9(2):11–19, 2004.

[2] James F Blinn. A generalization of algebraic surface drawing. *ACM transactions on graphics (TOG)*, 1(3):235–256, 1982.

[3] H. S. M. Coexter. *Introduction to Geometry, (pp. 93 and 289-290)*. Newyork Wiley, 1969.

[4] Ashish Gupta, George Allen, and Jarek Rossignac. Exact representations and geometric queries for lattice structures with quador beams. *Computer-Aided Design, 2019 (in press)*, 2019.

[5] Ashish Gupta, Kelsey Kurzeja, Jarek Rossignac, George Allen, Pranav Srinivas Kumar, and Suraj Musuvathy. Programmed-lattice editor and accelerated processing of parametric program-representations of steady lattices. *Computer-Aided Design*, 2019.

[6] John C Hart. Sphere tracing: A geometric method for the antialiased ray tracing of implicit surfaces. *The Visual Computer*, 12(10):527–545, 1996.

[7] Menelaos I Karavelas and Mariette Yvinec. Dynamic additively weighted voronoi diagrams in 2d. In *European Symposium on Algorithms*, pages 586–598. Springer, 2002.

[8] David L Millman. Degeneracy proof predicates for the additively weighted voronoi diagram. Master's thesis, Courant Institute of Mathematical Sciences New York, 2007.

[9] Evan D Nash, Ata Firat Pir, Frank Sottile, and Li Ying. Convex hull of two circles in r3. *Combinatorial Algebraic Geometry*, page 297, 2016.

[10] Barry Spain. *Analytic Quadrics, (pages 102-104)*. Pergamon Press, 1960.

[11] Vinod Srinivasan, Esan Mandal, Ergun Akleman, et al. Solidifying wireframes. In *Proceedings of the 2004 bridges conference on mathematical connections in art, music, and science*, 2005.

[12] David Stasiuk and Daniel Piker. Exoskeleton - bespoke geometry. `http://www.bespokegeometry.com/2014/05/17/exoskeleton/`. Accessed: 2019-25-01.

[13] Guanglei Xiong, Suraj Musuvathy, and Tong Fang. Automated structured all-quadrilateral and hexahedral meshing of tubular surfaces. In *Proceedings of the 21st International Meshing Roundtable*, pages 103–120. Springer, 2013.

Computing intersection areas of overlaid 2D meshes

W. Randolph Franklin [1] and Salles Viana Gomes de Magalhães [2]

[1] *ECSE Dept, Rensselaer Polytechnic Institute, Troy NY USA,* `https://wrf.ecse.rpi.edu/`
[2] *Departamento de Informática, Universidade Federal de Viçosa, MG, Brasil*

Abstract

Parover2 is a parallel algorithm and preliminary implementation to compute the area of every nonempty intersection of any face of one 2D mesh with any face from another mesh over an overlapping domain. This is the hard part of cross-interpolating data from one mesh to another, for when the faces one mesh have an attribute that would be useful for the faces of the other mesh. Parover2, implemented using a map-reduce paradigm in Thrust, can quickly process millions of faces. The expected execution time is linear in the number of intersections, which is usually linear in the number of input faces. A uniform grid quickly determines the pairs of input edges that might intersect. Local topological formulae compute the areas of the output faces from the sets of their (vertex, edge) adjacencies w/o needing to compute the faces' global topologies.

1 Introduction

Suppose that a polygon $\mathcal{P}$ has been meshed, or partitioned, into smaller faces, in two different ways, M_0 and M_1 for two different applications. Each mesh is optimal for some application, and would be suboptimal for the other application. Assume that the faces of M_0 have some property, such as mass, that would be useful for the faces of M_1. (If the density varies, then the mass is not simply the area.) One quick approximation for the mass of f', a face of M_1. is a weighted sum of the masses of the overlapping faces of M_0, with the weights being the areas of the intersections of f' with the overlapping faces of M_0. The compute-bound component is to identify all the nonempty intersections of any face of M_0 with any face of M_1, and compute their areas. This paper presents Parover2, an algorithm and preliminary parallel implementation for that.

Parover2's execution time is linear in the number of intersections, which is generally linear in the number of faces. An initial cull to locate pairs of faces likely to intersect is achieved with a uniform grid. Parover2 uses local topological formulae to compute the areas of the output faces (aka outfaces) from their vertices and half-edges. The algorithm is largely a series of map-reduce steps, implemented with Nvidia's Thrust, with an OpenMP backend.

3D-EPUG-Overlay, Magalhães et al [5, 11, 12, 13, 17] is a somewhat similar idea. Working in 3D, that finds the complete intersection polyhedra, where we find only areas. However it directly uses OpenMP, while we use Thrust, so our possible implementation platform will also include an Nvidia GPU. 3D-EPUG-Overlay also uses rational numbers to completely prevent roundoff errors. Because of the specialized nature of its output, Parover2 is much faster. However 3D-EPUG-Overlay handles geometric degeneracies with Simulation of Simplicity.

Common algorithms for computing intersections include plane sweep lines (in 2D) and quad or octrees [2, 20]. According to Audet [1], "the plane sweep strategy does not parallelize efficiently, rendering it incapable of benefiting from recent trends of multicore CPUs and general-purpose GPUs", and "While effort has been expended to parallelize the plane sweep on CPU, most recently by ([18, 19]), none of the proposed candidates result in an algorithm amendable to fine-grained SIMD parallelism such as with GPUs".

QuickCSG [6] is an efficient parallel intersection algorithm using a kd-tree index. However, it assumes vertices are in general position, and does not handle floating point numerical non-robustness [21], although the user may randomly perturb the input to help.

CGAL [3] operates on Nef polyhedra, which are boolean combinations of half-spaces. LibiGL [15] is an exact and parallel algorithm, Zhou et al.[21], for performing booleans on meshes. LibiGL is also much faster than CGAL for Nef Polyhedra, but is still slower than fast inexact algorithms such as QuickCSG.

The simpler problem of computing the area of the intersection of two polygons is useful in interference detection in robotics. However, the usual solution is either more complicated; it first computes the intersection polygon, and then its area, or it's simpler and tests only whether the intersection is nonempty.

While there appears little archival literature on directly computing intersection areas, some web pages exist. That is, while every CAD package can perform boolean operations on polygons and polyhedra, and then can compute mass properties, optimizing the composition of those two functions is rather rare. Nevertheless, Hardy [14] gives an algorithm with code for the area of the intersection. Polylib [16] operates on objects that are unions of d-D polytopes. Edwards [7] describes how to find the area of the intersection of two polygons with a graphing calculator.

2 Data structures

Parover2 reads the input meshes in one format, and computes the output mesh in a different, simpler, format. Each format is sufficient for the necessary computation. The input mesh format is designed to compute the outface vertices. The outface format is designed compute the outface areas.

The input mesh format is the set of its edges, $\{(x_0, y_0, x_1, y_1, f_l, f_r)\}$. $(x_0, y_0), (x_1, y_1)$ are the coordinates of one edge's end vertices. f_l and f_r are the identifiers (ids) of its adjacent faces to the left and the right. There are no lists, and no explicit vertices, explicit faces, or higher level topology such as the edges around a vertex, the edges around a face, nested faces, etc. That is, although several edges may contain the same vertex, we do not record this. We do not record in one place all the vertices contained in one face, etc. This info could be derived if we needed it, which we don't. This is simpler than quad edges and doubly connected edge lists, because it permits fewer operations (but it permits all the operations that we need).

The outface format is even simpler; it does not even store complete edges. It is a set of edge-vertex incidences, as described later. However it permits the local topological formulae described next to compute the outface areas.

3 Local topological formulae

This section will motivate the power of local topological formulae, by presenting an algorithm for computing the area of a face from either the set of its vertex positions and their neighborhoods, or the set of its vertex-edge incidences. The neighborhood of a vertex is defined as the direction vectors of its adjacent edges, and which one of the two sectors that they define is inside the face. This algorithm does not use the lengths of the adjacent edges, nor the vertices at their other ends. It resembles Green's theorem, except that, while Green's theorem computes a polygon's area by integrating along its boundary edges, we use only the boundary vertices.

The general formula is given in [8]. For a simple example, consider a diagonal rectilinear face f, as shown in Figure 1. All edges have slope $+1$ or -1. The face may have multiple components and nested islands and holes. Each vertex will be assigned a sign bit, s, determined by its neighborhood. Then the area of this isothetic diagonal f is $\sum_i s_i x_i^2$. For Figure 1, the area would be $0-1+4-9+16-4=6$. As this formula is a map-reduction, it may be efficiently computed in parallel.

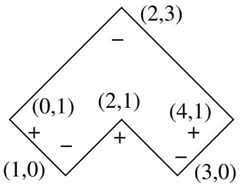


Figure 1: Diagonal rectilinear polygon

The proof is by induction. It is clearly true if f is a rectangle. If two rectangles with a common edge are united and those common edges (and four vertices) removed, then the area formula remains valid. In this way, any face may be built up. Any mass property, such as a higher order moment, may be likewise computed.

There are many formulae using different input formats. Another face area formula map-reduces the set of oriented edges, summing the signed areas subtended by the edges and the coordinate origin.

We compute the outface areas with a formula using the set of *(vertex, edge)* adjacencies. Each vertex neighborhood comprises two adjacencies. For vertex v with adjacent edges e_1, e_2, they would be $(v, \widehat{e_1})$ and $(v, \widehat{e_2})$. The normalization is because we know the direction of the edge but not its length. One adjacency is represented as the triple $(v, \widehat{t}, \widehat{n})$. v is the position of the vertex. $\widehat{t}$ is a unit direction vector along the edge. $\widehat{n}$ is a unit direction vector perpendicular to $\widehat{t}$ pointing to the inside side of the edge. $\widehat{n}$ adds only one bit of information.

The area of the general f is $\left(\sum \left(v \cdot \widehat{t}\right)\left(v \cdot \widehat{n}\right)\right)/2$. This is proved by dropping a perpendicular from the origin to each edge of f, and partitioning f into $2n$ right triangles. The vertices of one triangle will be the origin, a perpendicular foot, and one of the two end vertices of that foot's edge. Then, the signed area of one such triangle is $\left((v \cdot \widehat{t})(v \cdot \widehat{n})\right)/2$.

The above data structure is simpler, smaller, and faster than the well known half-edge data structure of the doubly connected edge list, since we do not use both ends of an edge together. It is much simpler to compute outfaces in this format than to also compute their edges.

4 Outface area computation strategy

Each outface is the intersection of two input faces (aka infaces), one from each input mesh. An outface is identified by the ordered pair of those two infaces. PAROVER2 computes the outface areas by a map-reduction over the vertex-edge adjacencies. As it processes the input, it does not compute each outface all at once. Rather, it will compute an output vertex with all its vertex-edge adjacencies, then another output vertex with all its vertex-edge adjacencies, and so on. So, it accumulates the outface areas incrementally. It never computes the output edges. (If necessary, the adjacencies could be paired up to produce the output edges.)

There are two types of output vertex-edge adjacencies: an adjacency of one of the input meshes M_i, and an intersection of an edge of M_0 with an edge of M_1.

4.1 Output adjacencies that are input adjacencies

1. Call the input adjacency, h.
2. Without loss of generality, assume that h is in mesh M_0.
3. h is adjacent to two infaces, call them f_l and f_r.
4. Let the vertex of h be v. v is contained in some face, say f', of the other mesh, M_1.
5. h is part of the two outfaces (f_l, f') and (f_r, f').
6. The normal vector component of h may need to be negated depending on which outface we are considering.
7. PAROVER2 will compute an area component for the two outfaces. Each component is of the form *(outface-id, area-component)*.
8. The total area of each outface is obtained by summing its components, as described later.

There are three nontrivial parts here: the tedious process of getting the several different special cases correct, storing the area components, and point location. Storing the area components is complicated. We are computing and storing the components in parallel. We do not know in advance the ids of the nonempty outfaces. We do not know in advance how many components each outface will have (aka how many vertices it has). PAROVER2 uses a vector of the *(outface-id, area-component)* pairs as follows.

1. The size of the vector is four times the number of input edges in the two meshes combined.
2. The i-th input edge will create output pairs numbered $4i$ to $4i+3$. So, the output pairs can be written in parallel w/o needing semaphores or locks. Specifically, we define a function that maps from index i to output pair i, and then map that function over the index vector $0, 1, 2, \ldots$.
3. Finally we sort the vector by outface-id and perform a parallel reduce-by-key, which sums the area components with the same outface-id.
4. The slowest step is the sort. However, every alternative, such a hash table keyed by the outface-id, or linked lists, appears worse.

For locating which face of M_1 contains v, we use a uniform grid, aka bucket sort. The expected query time is constant per point location, independent of the map size. The preprocessing algorithm goes as follows:

1. Determine the maximum linear size of any face in either direction (x or y).
2. Superimpose a grid of $g \times g$ cells over the meshes. g is chosen so that a cell is slightly taller and wider than the largest face.

3. For each input edge e, compute a superset of size 4 of the grid cells that intersect e. Because of the choice of g, the actual number of cells intersecting e ranges from 1 to 3. For programming ease, we circumscribe e with a box and use all the cells intersecting that box. This avoids exactly computing—in parallel—the intersecting cells, which would require identifying, in constant time, the i-th intersecting cell.
4. Working in parallel over the edges, form a vector of these pairs.
5. Sort it by cell id.
6. Use a parallel scan function to find the start of each cell's edges in the sorted vector, and the number of edges in each cell.

The total preprocessing execution time is linear in the input size because (a) for sorting by cell ids, a linear-time radix sort is applicable, and (b) the other steps are linear in the number of pairs (and fast).

The query algorithm to locate which face mesh M_1 contains point q from mesh M_0 is this:

1. Compute which grid cell contains q. (This is simply two modulo operations on q's coordinates.)
2. Process the edges of M_1 that are in the same cell as q as follows, in parallel over the edges, as follows. (a) Run a vertical line up and down from q through the cell. (b) Skip edges that are not intersected. (c) For each edge that is intersected, compute the vertical distance from q to the edge. (d) Return the absolutely closest edge.
3. If no edge was vertically above or below q, then the containing face is the external face. This is because the grid size is large enough that it is impossible to a face to extend beyond a cell in both directions.
4. Otherwise the containing face is one of the two faces adjacent to the closest edge. Which one is determined by whether the edge is above or below q and whether its first vertex is to the left or right of its second vertex.

The execution time to query q is linear in the number of edges in its grid cell. Since the cell size is slightly larger than the largest face, that depends on the ratio between the average face size and the largest face size, which is a constant < 4 for many common distributions of face sizes, including uniform and Gaussian. So, the expected query time per point is constant.

4.2 Output adjacencies that are the intersections of two input edges

This case differs from the previous case in two ways. First, we must compute the intersections of edges from M_0 with edges from M_1. Each intersection is a vertex of four outfaces and generates two adjacencies per face, for a total of eight outface adjacencies per intersection. Second, there is no need for point location because here we know the outface ids.

Note that two edges that intersect each other must both intersect the same grid cell, but not all pairs of edges in the same cell will intersect. The edge pairs in a cell can be indexed so that the i-th pair can be determined in constant time. If the edges are independently and identically distributed, then the probability of a pair of edges in the same cell intersecting each other is constant, independent of the total number of edges.

The algorithm goes as follows.

1. Compute the maximum possible number of edge intersections per cell and allocate space for a vector of intersections from all cells.
2. Populate that vector with the pairs of possibly intersecting edges. Note that this parallelizes because of the above observations. I.e., we can determine in constant time the i-th element of that vector.
3. Filter that vector by whether or not the edges do intersect.
4. Map the resulting vector into a vector of octuples of outface adjacencies.
5. Sort, reduce by key, and sum.

This parallelizes well. The expected execution time is linear in the number of input edges. An adversary could generate bad input cases, but other data structures such as quadtrees can also be made to perform poorly, and they don't parallelize well. We do not believe that real data would have this problem.

5 Implications of the target platform

The design choices in Parover2 are motivated by the goal of eventual execution on an Nvidia GPU. (Nvidia was chosen because it is currently by far the most common and most cost-effective GPU.) A GPU executes thousands of threads in parallel, with the threads grouped by 32 into warps. All 32 threads in one warp simultaneously execute the same instruction, ideally on data from successive words in memory. The only exception is that some threads in the warp may be idle.

Efficient algorithms for a GPU prefer data structures that are arrays of plain old data types, not even arrays of structures. Conditionals hurt performance. Pointers are strongly to be deprecated. Even randomly accessing elements is not ideal.

Therefore complex and adaptive data structures such as sweep lines and trees are very difficult to parallelize. This is especially true when using hundreds or thousands of threads. All this gets much worse in 3D. Hence our preference for simple flat data structures like uniform grids.

Simple flat data structures have another advantage; they are more compact; they take less space. In many parallel programs, the I/O time to read and write the data dominates the computation time. Although hosts and devices have a lot of memory as described above, it is slow, but they have small fast caches and register banks.

6 Implementation

Parover2 is implemented in C++ on a dual 14-core 2.0GHz Intel Xeon with 256GB of memory, running Ubuntu Linux. The parallel environment is Nvidia's Thrust, which is a set of parallel C++ classes and routines that map and reduce vectors. It is a parallel extension of parts of the Standard Template Library and Boost[4]. Thrust is reasonably mature, and seems to represent a good medium being a high level abstraction and being efficient. Thrust has several possible backends, including OpenMP and CUDA. Parover2 currently uses OpenMP, with CUDA support being debugged.

Our initial test cases are pairs of overlapping square meshes. Times are shown in Table 1. The test times start after the data has been read. The additional time to read the data, from ASCII files stored in real memory, is about 80% of the processing time. If the identical job is rerun, the time may vary by 10%. The parallel speedup on this system with 28 threads and 56 hyperthreads was a factor of 6.3. One limiting factor is that the processors automatically overclock from 2.0GHz to 3.2GHz when lightly loaded, but run more slowly when executing many parallel threads. This is typical of multicore CPUs, and serves

No. input edges	No. output faces	Elapsed time (sec)
220	400	.023
3,720	3,600	.032
40,400	40,000	.082
361,200	360,000	.47
4,004,000	4,000,000	6.2

Table 1: Elapsed time to compute intersection areas of two meshes

to limit the heat. The CPU time increased massively with the parallelism; for 56 threads it was 198 CPU seconds. This is irrelevant because the usual metric for parallel programs is elapsed wall-clock time.

7 Extension to 3D

The goal of this project is to compute the volumes of all the intersecting regions from two overlaid meshes in 3D. The mesh polyhedra may be tetrahedra, hexahedra, or other polyhedra. The conceptual extension is small; the difficulties mostly practical. Indeed, we expect that the tools that make PAROVER2 work, i.e., local topogical formulae and uniform grid, will be even more valuable in 3D, in PAROVER3.

Each input mesh for PAROVER3 is a set of faces, each tagged with the ids of its adjacent polyhedra. The output is the set of the pairs of input polyhedra that have a nonempty intersection, together with that intersection's volume.

Our volume formula for a polyhedron is a map-reduce over the set of 3D adjacencies. A 3D adjacency is a 4-tuple $(v, \widehat{t}, \widehat{n}, \widehat{b})$, where v is the position of a vertex, $\widehat{t}$ is a unit vector tangent to an adjacent edge, $\widehat{n}$ is a unit vector normal to $\widehat{t}$ and in the plane of a face adjacent to that vertex and edge, and $\widehat{b}$ is a unit binormal vector, normal to both $\widehat{t}$ and $\widehat{n}$ and pointing towards the interior of that face.

If v has k adjacent edges, then it will have $2k$ adjacencies. The volume of the polyhedron is $\left(\sum((v\cdot\widehat{t})(v\cdot\widehat{n})(v\cdot\widehat{b})\right)/6$. The unnecessity of any global topological info, even complete edges or faces, makes this formula easier to apply to the intersection of two 3D meshes.

Union3 [9, 10] is another parallel algorithm and implementation that demonstrates the validity and efficiency of these ideas. Union3 computes the volume of the union of millions of congruent isothetic cubes. When the cubes' positions are i.i.d., the expected execution time is linear in the number of cubes, even if the number of face-edge and face-face-face intersections grows superlinearly. The reason is that only those intersections that are not inside any input cube become output vertices. That number grows only linearly. When a cell is completely inside a cube, the possibly superlinear number of intersections inside it are never computed. A more detailed theoretical analysis, with implemention details and test results is given in the references.

References

[1] Samuel Audet, Cecilia Albertsson, Masana Murase, and Akihiro Asahara. 2013. Robust and Efficient Polygon Overlay on Parallel Stream Processors. In *Proceedings of the 21st ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems (SIGSPATIAL'13)*. ACM, New York, NY, USA, 304–313. https://doi.org/10.1145/2525314.2525352

[2] Mark de Berg, Otfried Cheong, Marc van Kreveld, and Mark Overmars. 2008. *Computational Geometry: Algorithms and Applications* (3rd ed.). Springer-Verlag TELOS, Santa Clara, CA, USA.

[3] CGAL. 2018. Computational Geometry Algorithms Library. (2018). Retrieved 2018-09-09 from https://www.cgal.org

[4] Beman Dawes, David Abrahams, and Rene Rivera. 2010. Boost C++ libraries. (2010). Retrieved 2018-09-09 from http://www.boost.org/

[5] Salles Viana Gomes de Magalhães. 2017. *Exact and parallel intersection of 3D triangular meshes.* Ph.D. Dissertation. Rensselaer Polytechnic Institute, Troy, NY, USA.

[6] Matthijs Douze, Jean-Sébastien Franco, and Bruno Raffin. 2015. *QuickCSG: Arbitrary and faster boolean combinations of n solids.* Ph.D. Dissertation. Inria-Research Centre, Grenoble–Rhône-Alpes, France.

[7] C. C. Edwards. 2018. How to Find the Area of the Intersection of Two Polygons. (2018). Retrieved 2019-03-18 from https://www.dummies.com/education/graphing-calculators/how-to-find-the-area-of-the-intersection-of-two-polygons/

[8] Wm Randolph Franklin. 1987. Polygon properties calculated from the vertex neighborhoods. In *Proc. 3rd Annu. ACM Sympos. Comput. Geom.* 110–118.

[9] W. Randolph Franklin. 2004. Analysis of Mass Properties of the Union of Millions of Polyhedra. In *Geometric Modeling and Computing: Seattle 2003*, M. L. Lucian and M. Neamtu (Eds.). Nashboro Press, Brentwood TN, 189–202.

[10] Wm. Randolph Franklin. 2005. Mass Properties of the Union of Millions of Identical Cubes. In *Geometric and Algorithmic Aspects of Computer Aided Design and Manufacturing, DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, Ravi Janardan, Debashish Dutta, and Michiel Smid (Eds.). Vol. 67. American Mathematical Society, 329–345.

[11] W. Randolph Franklin, Salles V. G. Magalhães, and Marcus V. A. Andrade. 2017. 3D-EPUG-Overlay: Intersecting very large 3D triangulations in parallel. In *2017 SIAM conference on industrial and applied geometry.* Pittsburgh PA USA. (talk).

[12] W. Randolph Franklin, Salles V. G. Magalhães, and Marcus V. A. Andrade. 2017. An exact and efficient 3D mesh intersection algorithm using only orientation predicates. In *S3PM-2017: International Convention on Shape, Solid, Structure, & Physical Modeling, Shape Modeling International (SMI-2017) Symposium.* Berkeley, California, USA. (poster).

[13] W. Randolph Franklin, Salles V. G. Magalhães, and Marcus V. A. Andrade. 2018. Exact fast parallel intersection of large 3-D triangular meshes. In *27th International Meshing Roundtable.* Alberqueque, New Mexico.

[14] Normal Hardy. 2018. Area of Intersection of Polygons. (2018). Retrieved 2019-03-18 from http://www.cap-lore.com/MathPhys/IP/

[15] Alec Jacobson, Daniele Panozzo, et al. 2016. *libigl: A Simple C++ Geometry Processing Library.* Retrieved 2017-10-18 from http://libigl.github.io/libigl/

[16] Vincent Loechner. 2010. PolyLib - A library of polyhedral functions. (2010). Retrieved 2019-03-18 from http://icps.u-strasbg.fr/PolyLib/

[17] Salles V. G. Magalhães, Marcus V. A. Andrade, W. Randolph Franklin, Wenli Li, and Maurício Gouvêa Gruppi. 2016. Exact intersection of 3D geometric models. In *Geoinfo 2016, XVII Brazilian Symposium on GeoInformatics.* Campos do Jordão, SP, Brazil.

[18] Mark McKenney, Roger Frye, Mathew Dellamano, Kevin Anderson, and Jeremy Harris. 2016. Multi-core parallelism for plane sweep algorithms as a foundation for GIS operations. *GeoInformatica* (2016), 1–24. https://doi.org/10.1007/s10707-016-0277-7

[19] Mark McKenney and Tynan McGuire. 2009. A Parallel Plane Sweep Algorithm for Multi-core Systems. In *Proceedings of the 17th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems (GIS '09).* ACM, New York, NY, USA, 392–395. https://doi.org/10.1145/1653771.1653827

[20] Robert Sedgewick. 1988. *Algorithms (2nd Ed.).* Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA.

[21] Qingnan Zhou, Eitan Grinspun, Denis Zorin, and Alec Jacobson. 2016. Mesh Arrangements for Solid Geometry. *ACM Trans. Graph.* 35, 4, Article 39 (July 2016), 15 pages.

Spline-based interactive object segmentation using SplineRNN

Ivar Stangeby[1], Oliver Barrowclough[1], and Georg Muntingh[1]

[1]*SINTEF Digital, Oslo, Norway*

Abstract

Recent advances in deep learning have led to super-human performance on various tasks in image analysis. In particular, the spatial and temporal weight-sharing in convolutional and recurrent neural network architectures has allowed for efficient feature learning, forming the basis for solving higher order tasks. Successfully solving these tasks relies on large, high-quality labeled datasets, typically created in a time-consuming and expensive process involving human annotators. Thus, there is increasing demand for methods that automate parts of the labelling process, in order to enable quicker and higher quality training data. In this work we extend two such recent approaches known as PolygonRNN *[2] and* PolygonRNN++ *[1] to a spline-based approach that we call* SplineRNN. *The benefits of* SplineRNN *with respect to* PolygonRNN *and* PolygonRNN++ *include higher order approximations and the need for fewer control points to represent smooth geometry, whilst maintaining simple methods for manipulating the geometry by a human user.*

1 Introduction

1.1 Machine learning

Following a standard definition of machine learning [5], "a computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P improves with experience E." As an example, a machine might learn from a large image dataset E, to perform classification, localization or segmentation tasks T with some accuracy P.

Supervised learning

Supervised learning is a machine learning paradigm where a model is trained to perform the task of predicting the labels y corresponding to input data x. From a probabilistic point of view, this amounts to estimating a conditional probability distribution $p(y \mid x)$ from a dataset of pairs $E = \{(x_i, y_i)\}_{i=1}^m$.

Recent developments in supervised learning have made it possible to achieve superhuman performance in various prediction tasks, by extracting knowledge from human experts and transferring this to algorithms in the form of neural networks.

Compared to traditional machine learning methods, state-of-the-art deep neural networks have the potential of achieving vastly better performance P. However, this potential comes at the price of longer training times, as well as an increased appetite for labeled data. Creating such large labeled datasets is a time-consuming and expensive endeavour. Hence, the success of deep learning relies on efficient methods for creating training data.

Segmentation

One type of labeled training data is images segmented according to instance or object. The corresponding image segmentation task is important in various application domains, for instance for labeling approaching vehicles in autonomous driving and identifying and marking tissue in medical images for the purpose of resection.

In many cases, segmentation is a difficult and ambiguous task. Recently proposed methods for semi-automatic object segmentation (PolygonRNN, PolygonRNN++) use predicted polygonal boundaries delineating object masks. The human annotator specifies a bounding box around the object to be segmented, and a bounding polygon is predicted. The use of a polygon facilitates easy correction by a human annotator.

1.2 Geometric modelling

Splines have long been a popular tool for geometric modelling for a number of reasons. From the geometric point of view, their modern representation as linear combinations of B-splines equips them with several desirable properties. This representation makes it possible to impose and directly control the smoothness as desired. Extra degrees of freedom can easily be inserted by either refining the geometry, raising the degree, or decreasing the smoothness. The underlying B-splines enjoy local support, making it possible to modify the geometric object locally without changing the object elsewhere. Spline curves lie in the convex hull of their control points, allowing for an intuitive behaviour of the spline, with respect to the control points. From the computational point of view, this property guarantees a numerically stable evaluation. Being composed of higher order polynomials on intervals of limited size, splines can approximate smooth data with a higher order of approximation. For predominantly smooth geometric objects, splines also provide a compact representation.

2 Method

2.1 Problem formulation

In image segmentation, one commonly used performance measure P is the *intersection over union* (IoU) of the predicted mask and the ground truth mask. This is defined in terms of sets as

$$\mathrm{IoU}(A, B) := \frac{|A \cap B|}{|A \cup B|}, \tag{2.1}$$

and gives a measure of how good an overlap there is between the predicted object mask, and the ground truth object mask. Note that $0 \leq \mathrm{IoU}(A, B) \leq 1$, attaining the lower bound only for disjoint objects and the upper bound only for identical objects.

Ideally, we would like to optimize the network for this performance measure directly. However, in the general case it is not clear how to differentiate the intersection over union map, making it not viable to train neural networks using the standard method of backpropagation.

Several options to direct optimization of IoU exist. By interpreting the IoU as a reward-signal, one may incorporate the IoU in the optimization in an approach based on reinforcement learning, as done in PolygonRNN++. Another approach is to limit the number of possible control point locations, and restrict control points to lie in a discretized output-grid superimposed over the image. By doing this, the task of control point prediction can be recast as a binary classification task. This discretization however comes at a loss of freedom in where the control points are allowed to go. This approach was used in PolygonRNN and yielded good results.

In SplineRNN we decide to adopt the latter approach. The output from the network at a given time step is a probability distribution over the possible control point locations. Using cross entropy, this probability distribution is compared to the ground truth. We train the network using cross-entropy loss.

2.2 Model architecture

In its current iteration, the proposed SplineRNN is a direct adaptation of the PolygonRNN architecture (see [3] for implementation details). It can be seen as consisting of two distinct components. A convolutional neural network (CNN) is used for encoding the input image into a set of features. These image features are then fed into a recurrent neural network (RNN). This RNN uses this information to predict a sequence of control points which delineates a spline. The network is depicted in Figure 1.

The convolutional neural network

The CNN-component consists of a modified VGG-16 [8] employing skip-connections from each layer of the VGG. The skip-connections are there to help the model retain information about the image at several spatial resolutions. These skip-connections are similar to the refinement-modules introduced in [6]. They make it possible to attain both object-level information from the upper layers of the network, as well as pixel-level information from the lower layers.

The recurrent neural network

The RNN-component of SplineRNN is a two-layer ConvLSTM [7] that at each time-step outputs a control point. Since a spline control polygon can be considered a “sequence in time”, the problem of predicting a control polygon can be seen as a spatio-temporal prediction problem, and this motivates the use of a ConvLSTM over a more classical fully connected LSTM. The ConvLSTM retains spatial information that is lost in the fully connected LSTM.

At any given time-step the RNN accepts as input the image features from the CNN, as well as the first and the two last predicted control points. Passing in the two previous control points helps the network infer a sensible direction in which the next control point is likely to be placed. By passing in the first control point, the network is able to know when the control polygon is closed, and it may stop predicting.

3 Experiments and results

In order to train the SplineRNN network, we need a data-set of ground truth splines. We generate cubic spline approximations to the ground truth polygons from the CityScapes [4] dataset. This gives us a set of cubic control points that we can compare our predictions against. The spline approximations are generated using a least-squares approach on B-spline bases with uniformly spaced knots. The approximations are performed iteratively until either the given tolerance is achieved, or the maximum number of iterations is reached. In the current implementation we used a tolerance of two pixels. The control points are generated in floating point arithmetic, but are snapped back to the pixel grid by rounding the floating point values to the nearest integer. In this way, the training data is input in exactly the same format as for polygons, only that the points are interpreted as spline control points rather than vertices of a polygon.

Any spline is parametrized in terms of a sequence of non-decreasing numbers. This sequence is called the *knot vector*. The choice of knot vector has a large impact on the shape of the resulting spline curve (see Figure 3). The current iteration of SplineRNN employs uniform knot vectors. We also restrict our attention to cubic splines in this first implementation.

Initial qualitative results of our method seem promising. A sample predicted spline and the corresponding ground truth spline can be seen in Figure 2.

4 Future work

As mentioned above, in the current implementation we restrict ourselves to cubic splines on uniform knot vectors. In future iterations we would like to include prediction of the knot vector corresponding to the predicted spline coefficients in the network architecture. By incorporating this parametrization of the spline curves directly in the network we can utilize the full set of geometric properties that splines have to offer.

The extension to non-uniform knot vectors also applies to the generation of the ground truth spline approximations. We expect better results by allowing non-uniform refinement with fewer control points required to approximate the geometry. By also including multiplicities in the knot vector, the continuity of the spline can be reduced, allowing kinks in the geometry to also be accurately reproduced. Uniform knots were chosen in order to obtain initial results, but they typically result in many more control points being generated than necessary, as can be seen in the sample ground truth spline in Figure 2.

We would also like to incorporate SplineRNN in an interactive setting, as done for PolygonRNN and PolygonRNN++.

References

[1] David Acuna et al. “Efficient interactive annotation of segmentation datasets with polygon-rnn++”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition.* 2018, pp. 859–868.

[2] Lluis Castrejon et al. “Annotating object instances with a polygon-rnn”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition.* 2017, pp. 5230–5238.

[3] Lluís Castrejón et al. “Annotating Object Instances with a Polygon-RNN”. In: *CoRR* abs/1704.05548 (2017). arXiv: `1704.05548`. URL: `http://arxiv.org/abs/1704.05548`.

[4] Marius Cordts et al. “The Cityscapes Dataset for Semantic Urban Scene Understanding”. In: *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR).* 2016.

[5] Thomas M. Mitchell. *Machine Learning.* 1st ed. New York, NY, USA: McGraw-Hill, Inc., 1997. ISBN: 0070428077, 9780070428072.

[6] Pedro H. O. Pinheiro et al. “Learning to Refine Object Segments”. In: *CoRR* abs/1603.08695 (2016). arXiv: `1603.08695`. URL: `http://arxiv.org/abs/1603.08695`.

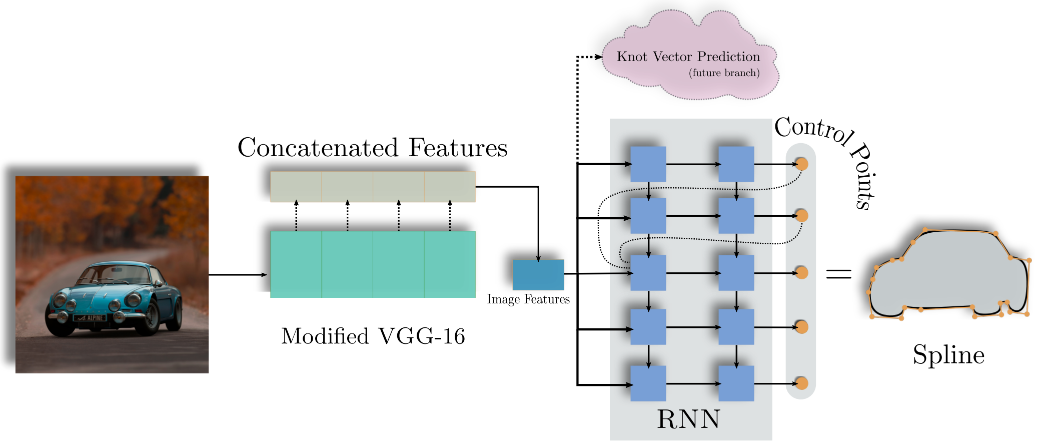


Figure 1: The proposed SplineRNN network architecture. An image is fed to the modified VGG-16, and a set of image features are extracted. These features are then passed as input into the RNN at each time-step, along with the first, and two previously predicted control points. The current iteration of the network does not feature the knot vector prediction branch. The network is therefore similar to the original PolygonRNN network. By incorporating knot vector prediction, we hope to be able to model sharper geometric features than what is made possible with the current rendition of the network.

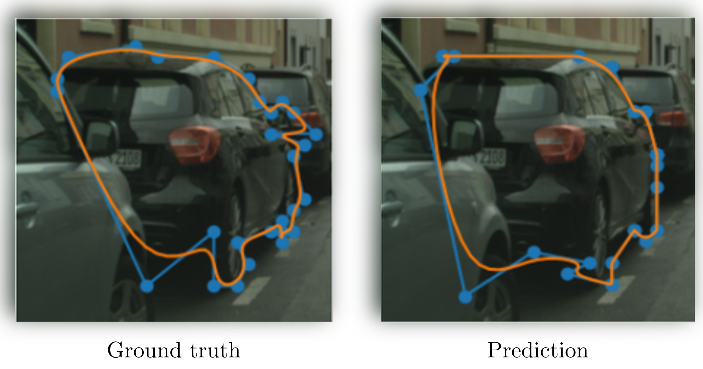


Figure 2: A sample prediction from the SplineRNN network. The predicted spline manages to follow the general outline of the car. It does however not manage to capture the small features like the outside rear-view mirror. Furthermore, note how the network places a fair amount of control points in order to get around curved features. This has to do with the parametrization of splines. In the current iteration of SplineRNN we are using uniform knot vectors for both the ground truth spline approximation, and the predicted splines.

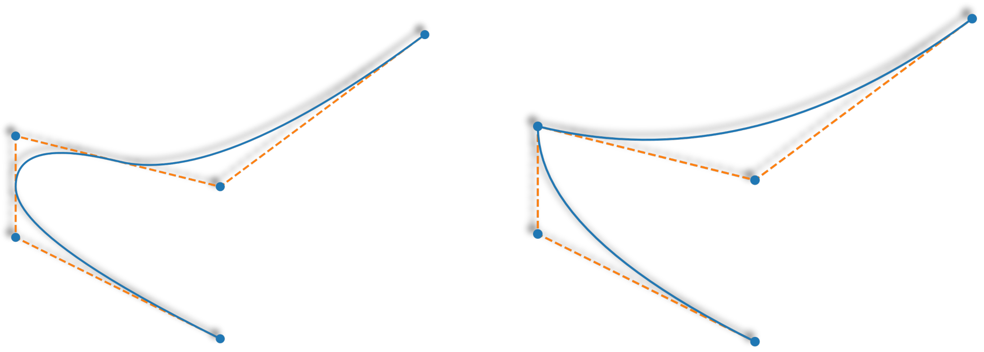


Figure 3: Two quadratic spline curves sharing the same control points but with different choices of knot vectors. The left curve has uniform knots, and the right curve has non-uniform knots. This image showcases the effect of choosing knot vectors on the resulting spline curve. By repeating knots, we are also able to model sharp creases in the geometry (as in the curve to the right). In the future, we would like SplineRNN to also predict the spline parametrization along with the control points of the spline. This will allow such sharp features to be modelled accurately.

[7] Xingjian Shi et al. "Convolutional LSTM Network: A Machine Learning Approach for Precipitation Nowcasting". In: *CoRR* abs/1506.04214 (2015). arXiv: `1506.04214`. URL: `http://arxiv.org/abs/1506.04214`.

[8] Karen Simonyan and Andrew Zisserman. "Very Deep Convolutional Networks for Large-Scale Image Recognition". In: *arXiv 1409.1556* (Sept. 2014).