GSO-Net: Grid Surface Optimization via Learning Geometric Constraints

Chaoyun Wang[1,2,3], **Jingmin Xin**[1,2,3], **Nanning Zheng**[1,2,3], **Caigui Jiang**[1,2,3*]

[1]National Key Laboratory of Human-Machine Hybrid Augmented Intelligence
[2]National Engineering Research Center of Visual Information and Applications
[3]Institute of Artificial Intelligence and Robotics, Xi'an Jiaotong University
chaoyunwang@stu.xjtu.edu.cn, {jxin, nnzheng}@mail.xjtu.edu.cn, cgjiang@xjtu.edu.cn

Abstract

In the context of surface representations, we find a natural structural similarity between grid surface and image data. Motivated by this inspiration, we propose a novel approach: encoding grid surfaces as geometric images and using image processing methods to address surface optimization-related problems. As a result, we have created the first dataset for grid surface optimization and devised a learning-based grid surface optimization network specifically tailored to geometric images, addressing the surface optimization problem through a data-driven learning of geometric constraints paradigm. We conduct extensive experiments on developable surface optimization, surface flattening, and surface denoising tasks using the designed network and datasets. The results demonstrate that our proposed method not only addresses the surface optimization problem better than traditional numerical optimization methods, especially for complex surfaces, but also boosts the optimization speed by multiple orders of magnitude. This pioneering study successfully applies deep learning methods to the field of surface optimization and provides a new solution paradigm for similar tasks, which will provide inspiration and guidance for future developments in the field of discrete surface optimization. The code and dataset are available at https://github.com/chaoyunwang/GSO-Net.

Introduction

The field of computer graphics is rife with discrete surface optimization problems, spanning applications such as 3D modeling, digital fabrication, physical simulation, and medical imaging. These problems encompass a variety of tasks such as the optimization of developable surfaces, surface flattening, and surface denoising (without emphasis, the term "surface" in this paper refers to "discrete surface" or "mesh" in general). Despite their different objectives, these tasks share a common foundation: optimizing certain geometric properties of the surface, such as flatness, smoothness, and curvature, while adhering to a set of geometric constraints such as surface continuity and boundary conditions.

Conventional surface optimization heavily relies on various numerical optimization techniques, including methods used in nonlinear optimization. While these methods have proven effective in dealing with small-scale or specific types of problems, they can struggle with large-scale and complex challenges, often encountering inefficiencies, convergence difficulties, and numerous hyperparameter issues (Nocedal and Wright 1999; Ma et al. 2021). In contrast, deep learning has made significant strides in the field of image processing. This success primarily stems from the regular structure of image data, allowing for specialized computations using convolutional kernels. These kernels perform sliding window computations, leading to efficient extraction and utilization of image features. This facilitates learning complex patterns from large amounts of data, an approach that might offer insights or alternatives to conventional optimization methods on discrete surfaces, especially grid surfaces in this paper. As (Yuan, Cao, and Shi 2023) mentioned in their overview of developable surfaces, there is scant research in the area of artificial intelligence applications to developable surfaces, they suggest building large-scale datasets for developable surface optimization.

In recent years, the application of deep learning in mesh processing has primarily centered on tasks like classification, segmentation, retrieval, and denoising, with a particular focus on surfaces represented by irregular triangular meshes. To handle such irregular structured data, two main strategies are adopted: designing learning networks suitable for irregular structured data or transforming the data into regular structured data. The former, represented by MeshCNN (Hanocka et al. 2019) and SubdivNet mesh processing networks (Hu et al. 2022), can process directly on triangle patches and are mainly used for semantic segmentation and recognition problems. However, they are not applicable for surface optimization problems. In the latter research involving regularized data processing, some studies parameterize the mesh to create geometric images for processing (Gu, Gortler, and Hoppe 2002). These studies tend to focus more on mesh reconstruction and compression (Sinha, Bai, and Ramani 2016; Wang et al. 2018; Ren et al. 2023). Also, there is work involving sampling on mesh patches to build regular structured data, which is common in surface denoising tasks (Wang, Liu, and Tong 2016; Shen et al. 2022; Zhao et al. 2022). While these representations can be processed with standard deep learning algorithms, from a data transformation perspective, they are lossy and cannot meet the precision requirements of tasks such as surface optimization.

*Corresponding Author.

From the current research landscape, it is clear that existing network architectures cannot be directly applied to surface optimization problems using deep learning methods. This is mainly due to the complex, irregular connectivity of triangular mesh representations, which are difficult for current deep learning algorithms to handle with high precision.

Inspired by the work of (Jiang et al. 2020, 2021), on quadrilateral mesh surfaces, we have found that quadrilateral meshes can be better defined and constrained in surface optimization tasks such as developable surface optimization and surface flattening, compared to triangular meshes. We noticed that the structure of regular quadrilateral meshes has a natural similarity to image data structure. As shown in Figure 1, we encode the vertex position information of Grid surface into RGB color information of image pixels. This process does not lose any information, both can be equivalently replaced and are well-suited for lossless compression and data processing.

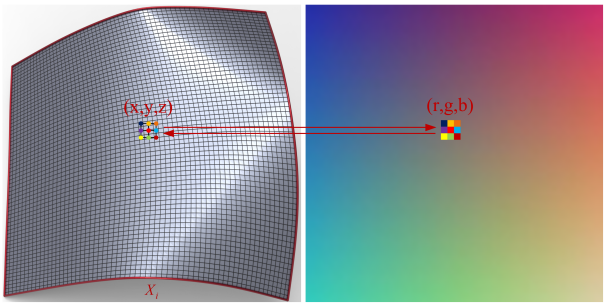

Figure 1: A grid surface and its corresponding geometric image.

This method allows for a lossless transformation of surface information, thus turning the surface optimization problem into an image processing problem. This conversion fits the prevalent deep learning image processing methods. Additionally, we can use the predefined geometric constraint properties in surface optimization as loss function for the network. This allows the network to learn surface optimization in a self-supervised manner, greatly improving the model's generalization capabilities and task versatility, and offers the potential to surpass numerical optimization algorithms in optimization results.

In summary, the main contributions of this paper are as follows:

- We exploit the data structure similarity between grid surfaces and images to encode grid surfaces as geometric images, using image processing methods to solve grid surface optimization problems.
- We have created a high-quality dataset of grid surface and corresponding results optimized by numerical optimization methods for research testing and comparison.
- We design a grid surface optimization network to learn geometric constraints in a self-supervised manner, which has been validated in several tasks for its generality and effectiveness. Compared with the traditional numerical optimization method, the optimization speed is improved by multiple orders of magnitude.

Grid Surface Dataset

For deep learning methods, datasets with large and diverse samples are crucial. In view of the fact that there are no relevant datasets available for surface optimization, we investigate the construction method for the grid surface optimization dataset used in this paper.

In surface construction, we utilize the Bezier surface construction algorithm proposed by (Aumann 2003). This method allows us to build surfaces of various shapes by using different quantities and arrangements of 3D control points, along with different degrees of Bezier curves. The grid surface is subsequently obtained by sampling 64×64 points on continuous Bezier surfaces.

To diversify the surfaces in the dataset, we focus on two essential features of surfaces, Gaussian curvature and shape, and ensure that surfaces in each feature interval are included in the dataset through a surface generation and screening procedure. Specifically, in terms of Gaussian curvature features, we use $K_P(M)$, $K_N(M)$, $K_A(M)$ to characterize surface M:

$$\begin{aligned} K_P(M) &= \tfrac{1}{n}\textstyle\sum_{k\in M} max(\kappa_G(p_k), 0) \\ K_N(M) &= \tfrac{1}{n}\textstyle\sum_{k\in M} min(\kappa_G(p_k), 0) \\ K_A(M) &= \tfrac{1}{n}\textstyle\sum_{k\in M} |\kappa_G(p_k)| \end{aligned} \tag{1}$$

where n represents the number of vertices of the surface M, $\kappa_G(p_k)$ is the discrete Gaussian curvature at vertex p_k computed with reference to (Meyer et al. 2003). In terms of shape features, we use σ^2 to characterize surface M:

$$\sigma^2 = \frac{\sum_{i=1}^{4}(X_i - \bar{X})^2}{4} \tag{2}$$

where X_i denotes the four boundary lengths of the grid surface as shown in Figure 1, $\bar{X}$ is the mean value of these boundary lengths.

Following the above approach, we generate 10401 grid surfaces for deep learning training and testing, moreover, include the results obtained using the optimization algorithm proposed by (Jiang et al. 2020) on the developable surface optimization and surface flattening tasks as a reference for traditional numerical optimization (TNO) method. Details of the dataset are given in the supplementary material.

Method

In image processing tasks, digital images are usually encoded using 8 bits, requiring only 1/256 accuracy. However, higher accuracy is required when expressing spatial grid vertex positions in the form of image data storage. In our experiments, we found that to ensure the basic smoothness requirements of the surface, for example, with a grid size of 64×64, the required precision needs to reach approximately 1/12500, which is about 50 times higher than image precision. This places demanding demands on the learning capabilities of deep learning networks. Moreover, unlike image data, which only represents color information, grid vertices

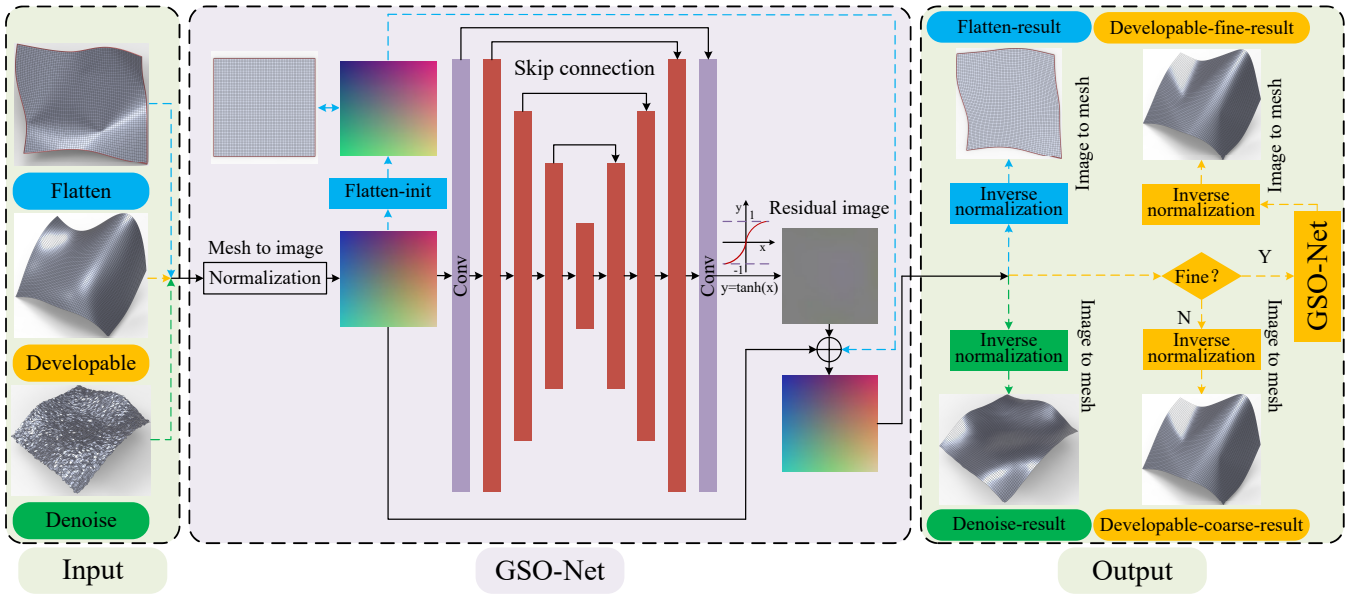

Figure 2: The pipeline of grid surface optimization. Different colored dashed lines are used in the Input and Output boxes to distinguish surface optimization tasks, and the middle GSO-Net box can be used for these tasks in general.

represent actual spatial coordinate information. After undergoing any orthogonal transformation, the information represented by the surface remains unchanged. Thus, this image encoding method also needs to take into account the invariance of the surface representation under orthogonal transformations. All these challenges call for the careful design of appropriate loss functions and network architectures to address them.

Grid Surface Optimization

In designing the grid surface optimization network, we draw on the image encoding-decoding structure commonly employed in deep learning networks. Our aim is to learn the mapping of the vertices of the grid surface in an end-to-end manner. Building on this design concept, we relate this task to the image denoising, where the noise can be understood as the displacement of vertices for surface optimization.

Figure 2 illustrates our proposed grid surface optimization pipeline for different tasks. Within the Input box, the three tasks we design are indicated by different colored dashed lines, and the same differentiation is applied in the Output box. The GSO-Net box in the middle represents the designed grid surface optimization network, which can handle these surface optimization tasks generically.

Specifically, in the pipeline for the developable surface optimization task, the input is a non-developable surface whose vertex positions are read and normalized before being encoded into a geometric image data format. Then, with the designed GSO-Net, a residual image representing the vertex displacement of the surface is obtained. By concatenating the residuals with the input image, we obtain an optimized geometric image representing a developable surface. Once the developability requirements are met, the developable surface can be directly output. For surfaces with high values of the Gaussian curvature introduced as input, fine networks can be employed to repeat the network operations described above, allowing for further optimization that ultimately results in a refined developable surface.

In the surface flattening task, a similar process is followed as in the developable surface optimization task. The difference lies in ensuring that the output is planar. Thus, the number of channels in the network output is set to 2. In GSO-Net, the blue dashed line in the input-output residual structure involves a Flatten-init for surface flattening, where the output represents the displacement of the initial mesh vertex.

In the surface denoising task, the processing flow is similar. The input is a noisy surface and the output is a smooth surface with noise filtered out. By GSO-Net, the basic noise features are learned and denoised with the residual structure.

Here is a specific explanation of the network module details mentioned in Figure 2:

Network Architecture In the GSO-Net, the "Conv" bar represents initial convolutional layer, and the red bar represents the feature extraction module, referring to the IMDB module in the image denoising network (Hui et al. 2019), skip connection is used between codec structures, the specific parameters are analyzed in the experimental section.

Normalization and Inverse Normalization Before feeding the surface data into the network, we apply a preprocessing method (Qi et al. 2017) that is suitable for point clouds rather than images to normalize the data, and we need to use these normalization parameters to inverse normalize the data for output.

Flatten-init In order to ensure that the initialization plane is as close as possible to the original surface flattening results and reduce the difficulty of network optimization, we

design an initialization method to ensure that the average mesh edge length of the 2d plane and the 3d surface in the sampling direction is equal.

Residual Image In the design, we refer to (Zhang et al. 2017) image denoising network and utilize the residual learning structure between input and output. This structure allows the network to focus on learning the displacement of mesh vertices between the input surface and the optimized surface, which facilitates learning and convergence.

Tanh Activation Function To avoid oscillations during training, we limit the range of network outputs. Inspired by (Zhang et al. 2020) approach in point cloud networks, we apply a tanh activation function to the output, restricting the vertex movement within the range of [-1, 1].

Coarse-to-Fine Coarse-to-fine is a computational strategy commonly used in the field of computer graphics. In the optimization of developable surfaces, considering the complexity of the surfaces, the network's learning capacity, and precision, we design a similar network as the mesh denoising networks by (Wang, Liu, and Tong 2016; Zhao et al. 2019; Shen et al. 2022), employing a cascaded optimization approach. We adopt a similar training and optimization strategy by using coarse and fine networks, which results in a two-stage optimization process for the surface, leading to better optimization performance.

Loss Functions

During the training of the grid surface optimization network, we exploit geometric constraints and prior knowledge related to surface optimization, transforming them into a form that is computationally convenient for geometric image processing. Here, the input geometric image is denoted by I, the optimized output geometric image by O, and the number of vertices for each grid by N. Additionally, c represents the number of image channels.

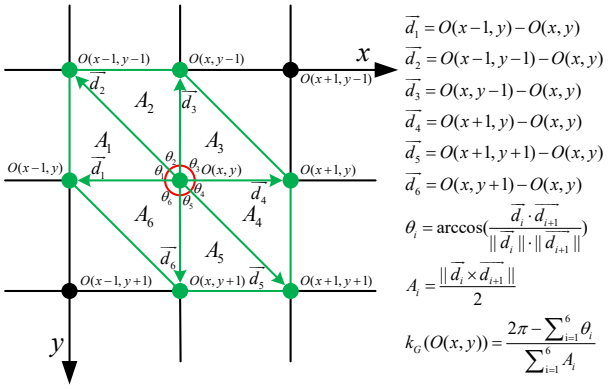

Figure 3: Discrete Gaussian curvature computation on geometric images.

We refer to several geometric properties of the optimized developable surface to design the loss function: it is developable, smooth, and close to the original surface, and correspond to the loss functions $loss_{gc}$, $loss_{fair}$, $loss_{in}$ used for network training. The following is the specific introduction.

$loss_{gc}$ is based on the fact that the Gaussian curvature at each vertex of the developable surface is zero. Figure 3 illustrates a schematic representation of calculating the discrete Gaussian curvature on geometric images. To Calculate the curvature at point $O(x, y)$, one first needs to determine the angle θ_i formed by the adjacent edges met at the point and A_i, which represents the areas of the surrounding triangles. The discrete Gaussian curvature $k_G(O(x, y))$ is then calculated by referring to (Meyer et al. 2003).

In order to facilitate the optimized calculation of geometric image loss, the convolution kernel calculation is used as the calculation $loss_{gc}$. The constant convolution kernel $W_{gc(i)}$ designed as shown in Figure 4(a) can be used to convolve the output geometric image, thereby obtaining the edge vector $\vec{d_i}$ required for calculating $k_G(O(x, y))$ in Figure 3. The formulas for the convolution calculation of $\vec{d_i}$ and the loss function $loss_{gc}$ are presented below:

$$\begin{aligned} d_i^{(c)} &= \sum_m \sum_n W_{gc(i)}(m,n) * O^{(c)}(x+m, y+n) \\ \vec{d_i} &= (d_i^{(1)}, d_i^{(2)}, d_i^{(3)}) \\ loss_{gc} &= \sum_x \sum_y \mid k_G(O(x,y)) \mid \end{aligned} \tag{3}$$

$loss_{fair}$ is designed to ensure the smoothness of the output surface. We refer to the smoothness calculation used in geometric optimization where adjacent vertices are forced to be located along a straight line. The constant convolution kernel W_{fair}, as depicted in Figure 4(b), is designed to compute the $loss_{fair}$ on the geometric image:

$$\begin{aligned} f^{(c)}(x,y) &= \sum_m \sum_n W_{fair}(m,n) * O^{(c)}(x+m, y+n) \\ loss_{fair} &= \sum_c \sum_x \sum_y |f^{(c)}(x,y)| \end{aligned} \tag{4}$$

$loss_{in}$ is designed to make the optimized surface closely resemble the original surface. We utilize the commonly used mean square error (MSE) loss function in image processing to compute $loss_{in}$ on the geometric image:

$$loss_{in} = \sum_x \sum_y (O(x,y) - I(x,y))^2 \tag{5}$$

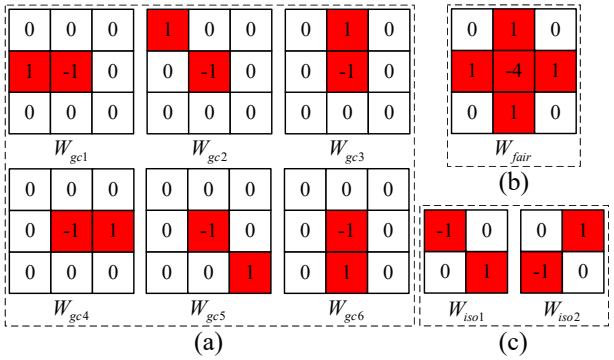

Figure 4: Constant convolutional kernel for computing geometric loss.(a) Gaussian curvature; (b) fairness;(c)isometry.

In the developable surface optimization task, the overall loss function $Loss_{developable}$ is as follows:

$$Loss_{developable} = w_{in} * loss_{in} + w_{fair} * loss_{fair} + w_{gc} * loss_{gc} \quad (6)$$

where w_{in}, w_{fair}, w_{gc} are the weighting coefficients for the corresponding loss function. By properly setting these weighting coefficients, the trained model can achieve favorable overall performance in terms of surface developability, proximity and smoothness.

In the surface flattening task, the term of isometry constraint defined in (Jiang et al. 2020) is used and converted to $loss_{iso}$ in geometric images, similarly to the calculation of $loss_{gc}$. The constant convolution kernels $W_{iso(i)}$ in Figure 4(c) are used to compute the quadrilateral diagonal vectors. This computation is performed in the same manner as the calculation of $\vec{d_i}$ in equation (3). The overall loss is as follows:

$$Loss_{flatten} = w_{iso} * loss_{iso} + w_{fair} * loss_{fair} \quad (7)$$

In the surface denoising task, our goal is to optimize the discrete surface so that it is similar to the input surface while ensuring fairness. To achieve this, we refer to the losses $loss_{in}$ and $loss_{fair}$, as described in equation (5) and equation (4). The overall loss is as follows:

$$Loss_{denoise} = w_{in} * loss_{in} + w_{fair} * loss_{fair} \quad (8)$$

Evaluation Metrics

We employ various evaluation metrics tailored to different tasks. Specifically, for the developable surface optimization task, we use proximity and developability evaluation metrics. In the surface flattening task, we apply isometry metrics, and for the surface denoising task, we utilize normal angle difference metrics. Let the sample size be denoted by n, and let the surfaces before and after optimization be represented as M and M'respectively, with N being the number of quadrilateral faces.

Proximity We indicate the shape similarity by the average Hausdorff distance d_H , with respect to the length of the bounding box diagonal, and the average evaluation metric on the dataset is d_{H-a}:

$$d_{H-a} = \frac{1}{n}\sum_{i=1}^{n} d_H(M_i) \quad (9)$$

Developability We evaluate the performance of surface developability optimization using K_{A-r}, which is the rate of surface Gaussian curvature, and the overall average evaluation metric on the dataset is K_{A-r-a}:

$$K_{A-r}(M) = (K_A(M) - K_A(M'))/K_A(M)$$
$$K_{A-r-a} = \frac{1}{n}\sum_{i=1}^{n} K_{A-r}(M_i) \quad (10)$$

Isometry We use the isometry loss on each quadrilateral cell as the evaluation metric, the formula is as follows:

$$loss_{iso-cell} = loss_{iso}/N \quad (11)$$

Normal Angle Difference We refer to the average normal angle difference θ, which is commonly used in mesh denoising tasks. We calculate the mean values $\theta_{noisy-a}$ and θ_{opt-a} before and after optimization on the dataset, and compute the average rate defined by $\theta_{opt-r-a}$ as the evaluation metric:

$$\theta_{noisy-a} = \frac{1}{n}\sum_{i=1}^{n} \theta(M_i)$$
$$\theta_{opt-a} = \frac{1}{n}\sum_{i=1}^{n} \theta(M_i') \quad (12)$$
$$\theta_{opt-r-a} = \frac{1}{n}\sum_{i=1}^{n} (\theta(M_i) - \theta(M_i'))/\theta(M_i)$$

Experiments and Results

In the experiments, we first analyze and select specific parameters of the GSO-Net using the developable surface optimization task as a reference, and then conduct sufficient experiments with this general network for the three surface optimization tasks mentioned above. Finally, the experimental results of the proposed method and traditional numerical optimization methods are comprehensively analyzed, both quantitatively and qualitatively. More details of the experiments and results are given in the supplementary material.

Developable Surface Optimization

In the developable surface optimization task, we initially use the optimization results of traditional numerical optimization methods as labels. This allows networks with different parameters to learn developable surface optimization in a supervised learning manner, enabling us to compare and select appropriate baseline network architectures. Subsequently, we compare the results of network learning in a self-supervised manner with different optimization strategies, and compare with traditional numerical optimization methods. Several surface optimization comparison example results are also presented.

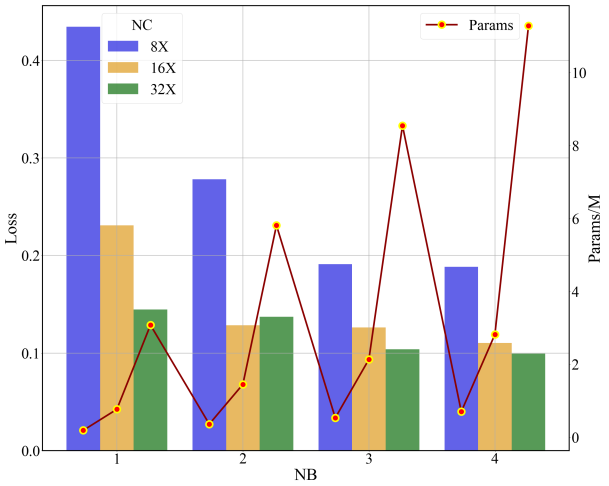

Figure 5: Comparison of optimization loss and parameter size for different network architecture parameters.

Network Architecture Parameters For our task, we determine the number of IMDB (NB) and channels in the feature maps (NC) in each layer of the network by examining

their fitting effect on the surface optimization. Moreover, the size of model parameters is considered as a reference criterion. The specific experimental results are shown in Figure 5. When NC was set to 16X (16,32,64,128) and NB was set to 4, we achieved perfect learning performance with a parameter size of only 2.82M. It can be observed that setting NC to 32X does result in a slight improvement in performance, but this comes at the cost of a significant increase in the parameter size.

Experimental Result Comparison In the experiment, we compare the traditional numerical optimization method with the GSO-Net method proposed in this paper, and verify the effectiveness of the coarse-to-fine optimization strategy. Assuming that the developability metric obtained by numerical optimization methods is already excellent, we adjust the optimization parameters of the model to make $loss_{gc}$ close to the result of the numerical optimization method. Then, we use the proximity metric to evaluate and compare the performance between the different methods.

Methods	TNO	Net-S	Net-C	**Net-CF**
$loss_{gc} \downarrow$	0.0283	**0.0273**	0.0417	0.0291
$K_{A-r-a} \uparrow$	**0.8888**	0.8776	0.8300	0.8768
$d_{H-a} \downarrow$	0.823%	0.608%	**0.414%**	0.517%
$time \downarrow$	38.256s	**0.013s**	**0.013s**	0.026s

Table 1: Developable surface optimization task evaluation metrics statistical results for different methods.

Table 1 shows the developable surface optimization results. In the "Methods" row, "TNO" represents the traditional numerical optimization method, "Net-C" and "Net-CF" represent the coarse and fine models in the coarse-to-fine strategy for GSO-Net, respectively, and "Net-S" represents the single model without this strategy.

In our experiments, the $loss_{gc}$ of the pre-optimized dataset is around 0.307. As shown in Table 1, all the methods reduce $loss_{gc}$ by about 91% (except Net-C as an intermediate model), and the performance of these methods on K_{A-r-a} is also relatively consistent.

Compared with TNO, the methods based on GSO-Net show improvements in terms of d_{H-a}. Specifically: (1) Net-CF reduces d_{H-a} from 0.8238% to 0.517%, a relative decrease of 37.24% compared to TNO; (2) Even without using a coarse-to-fine optimization strategy, Net-S reduces d_{H-a} from 0.823% to 0.608%, a relative decrease of 26.12%; (3) If the requirement for the Gaussian curvature value of the developable surface is not overly stringent, Net-C reduces d_{H-a} from 0.823% to 0.414%, a relative reduction of about 50%, while K_{A-r-a} is reduced by only 5.88%.

Comparing the results of Net-S and Net-CF, it can be observed that the proposed coarse-to-fine strategy can reduce the d_{H-a} from 0.608% to 0.517%, balancing developability and significantly enhancing the similarity between the surfaces before and after optimization.

The last row compares the time consumption among different methods. Since the proposed method can transform the surface optimization problem into an image processing problem, the speed of the proposed method is improved by multiple orders of magnitude compared with TNO.

The results in Table 1 represent the average optimization performance over all surfaces, including surfaces with various values of Gaussian curvature. In Figure 6, the optimization results for surfaces with different ranges of Gaussian curvature values are displayed. In this figure, the X-axis corresponds to surfaces with varying Gaussian curvature values, and the left and right Y-axes represent K_{A-r-a} and d_{H-a}, respectively.

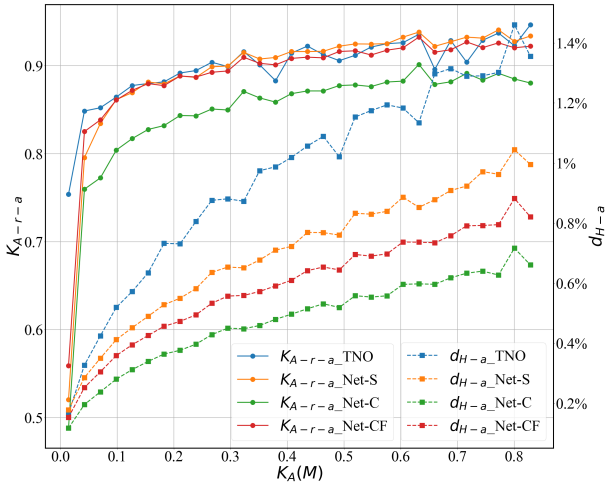

Figure 6: Line plots of the K_{A-r-a} and d_{H-a} statistics in different surface Gaussian curvature interval values.

From Figure 6, the K_{A-r-a} correlation curves of the proposed methods and TNO are observed to be closely aligned, corresponding to the K_{A-r-a} values in Table 1. Compared to TNO, the difference in d_{H-a} between the proposed method and TNO grows rapidly as the Gaussian curvature of the surface increases. This demonstrates that the proposed method is significantly more adept than TNO at handling complex surface optimization challenges. Additionally, the advantage of the proposed coarse-to-fine strategy is apparent in the correlation curves and corresponds to the results documented in Table 1. However, it's essential to recognize that when optimizing surfaces with particularly low Gaussian curvature values, or simple surfaces, TNO exhibits a higher K_{A-r-a} compared to the proposed method. This observation is in alignment with the inherent strengths and weaknesses of TNO, as delineated in the introduction.

In Figure 7, a collection of example results for developable surface optimization using different methods is exhibited. Accompanying each optimized surface, deformation heatmaps are provided. Within these heatmaps, the color is indicative of the value of d_H, representing the distance from the vertex to the original surface. Darker colors symbolize greater deformation in comparison to the original surface. A discernible observation from the heatmaps is that the coloration corresponding to the proposed method is considerably lighter than that associated with the traditional method. This visual evidence corresponds to the quantitative results

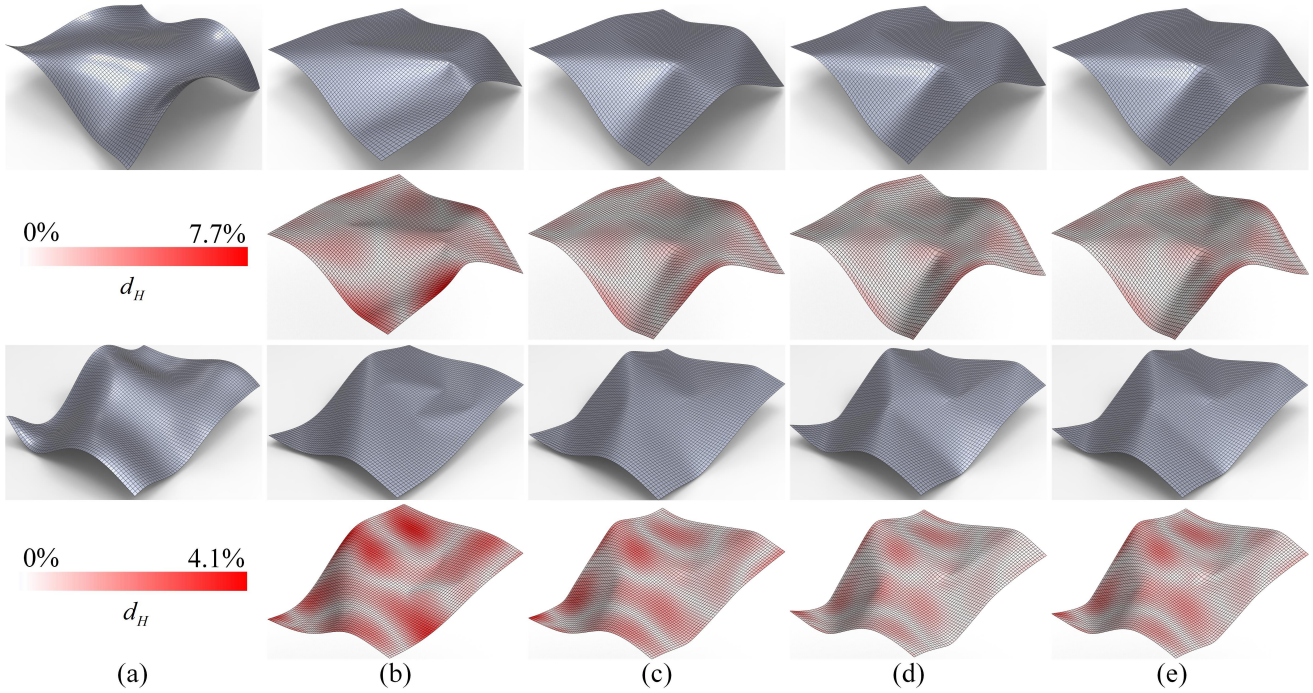

Figure 7: Comparison of example developable surface optimization results for different methods. (a) raw input surfaces; (b) TNO; (c) Net-S; (d) Net-C; (e) Net-CF.

presented in Table 1 for d_{H-a}. It provides a lucid demonstration that the proposed method has been successful in reducing surface deformation while simultaneously ensuring the development of the optimization results.

Surface Flatten

In the surface flattening task, we used the same data and networks as in the developable surface optimization task. Table 2 shows the results for different optimization methods.

In the context of Table 2, the "Methods" row details various strategies employed in this paper. The "Init" method, denoted as Flatten-init, serves as a baseline reference. "TNO" stands for the traditional numerical optimization approach commonly used in the field. The "Net" method introduces GSO-Net, our proposed solution. Building on the Net method, "Net-W" leverages prior knowledge to calculate the dynamic weight, w_{iso}, particularly in areas with high Gaussian curvature where the vertex itself is non-developable. This leads to a considerable isometry loss. The value of w_{iso} at the output point O(x,y) is equal to the normalized Gaussian curvature value of the point and subtracted by 1:

$$w_{iso}(x, y) = 1 - normalization(|k_G(O)|)(x, y) \quad (13)$$

Methods	Init	TNO	Net	**Net-W**
$loss_{iso-cell} \downarrow$	6.21e-3	3.30e-4	2.94e-4	**2.46e-4**
$time \downarrow$	**0.002s**	4.235s	0.015s	0.015s

Table 2: Surface flattening task evaluation metrics statistical results for different methods.

As can be seen from Table 2, compared to the baseline Init, most methods achieve a loss reduction rate of about 95%. Our proposed method Net outperforms the traditional method TNO, with Net-W further reducing losses by 16.32% relative to Net and 25.45% compared to TNO. Moreover, in the last row, the proposed method can reduce the time consumption by multiple orders of magnitude.

Figure 8 displays heatmaps of several surface flattening optimization results, where the color corresponds to the $loss_{iso-cell}$. The variation in color depth among the different methods is consistent with the results in Table 2. Notably, the contours in Net-W are closer to those in TNO compared to Net, showing that TNO's method provides better contouring. However, Net demonstrates superior performance in capturing detail. Net-W effectively combines the strengths of both to achieve the best overall results.

Surface Denoise

To further verify the generality of the proposed method, we conduct simple experiments on the surface denoising task. We add random noise to the original surfaces as noise surface input to the network for training and learning denoising. The surface noise addition is shown as follows:

$$M_{noisy} = M + N(\mu, \sigma)$$

where $N(\mu, \sigma)$ represents Gaussian noise, where μ= 0 is set and different levels of noise are generated by adjusting the parameter σ. Considering that the grid resolution is about 1/64, we set the maximum value of σ to 0.015 and the minimum value to 0.001 to study the effect of surface noise removal under different noise levels.

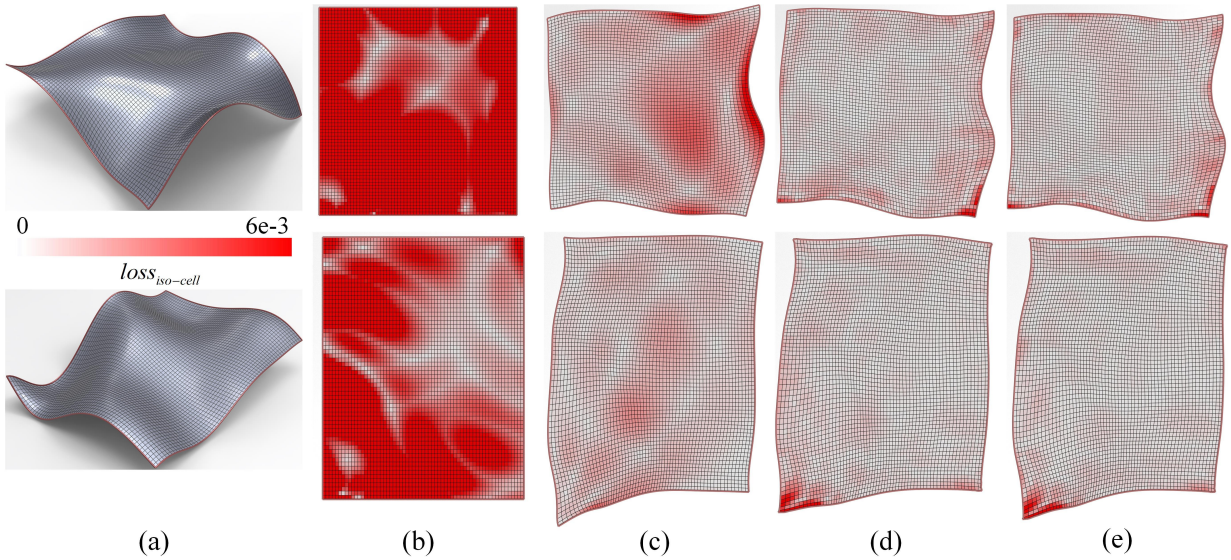

Figure 8: Comparison of example surface flattening results for different methods. (a) Raw input surfaces; (b) Init; (c) TNO; (d) Net; (e) Net-W.

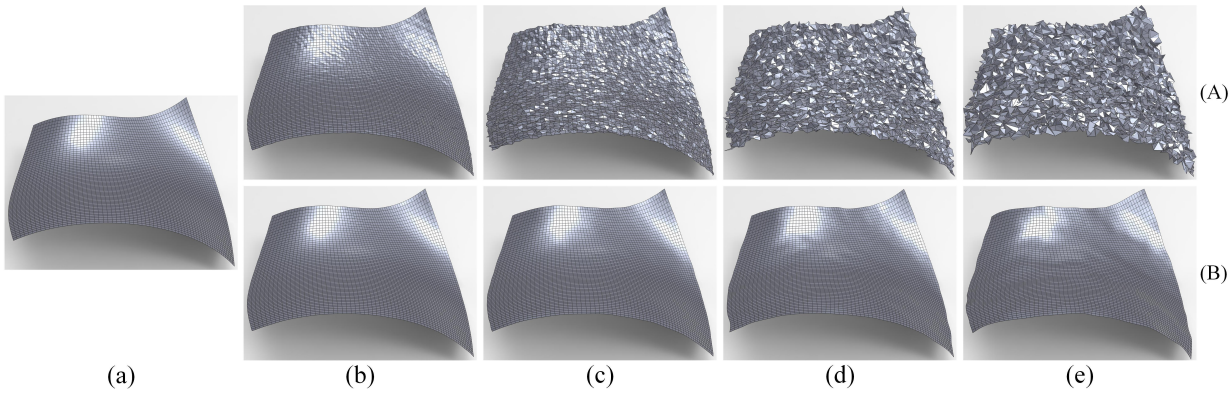

Figure 9: Comparison of example surface denoising results for different noise levels. (A) noise surfaces; (B) denoising results; (a) raw smooth surface; (b) $\sigma = 0.001$; (c) $\sigma = 0.005$; (d) $\sigma = 0.010$; (e) $\sigma = 0.015$.

Noise level(σ)	0.001	0.005	0.010	0.015
$\theta_{noisy-a}$	4.02°	21.36°	44.64°	61.59°
θ_{opt-a}	0.633°	1.502°	2.505°	3.544°
$\theta_{opt-r-a}$	83.84%	92.85%	94.42%	94.30%

Table 3: Surface denoising task evaluation metrics statistical results for different noise levels.

Table 3 shows the statistical results of surface denoising tasks using GSO-Net. The results show that the proposed method can denoise surfaces with different noise levels very well, and the optimization efficiency does not decrease as the noise level becomes higher.

In Figure 9, a surface denoising example is presented. A visual examination reveals that the optimized results align with the statistical findings for θ_{opt-a}, as shown in Table 3. When compared with the original smooth surface, it's evident that the proposed method is capable of achieving excellent surface recovery for various levels of noise.

Conclusion

In this work, we introduce an innovative approach to encode the vertex location information of a grid surface into a geometric image, and subsequently address the grid surface optimization problem through the use of image processing methods. To implement this, we constructed a high-quality grid surface dataset and designed a grid surface optimization network, GSO-Net, applicable to general tasks. This network employs geometric constraint loss for self-supervised learning and can be effectively used for developable surface optimization, surface flattening, and surface denoising tasks. Experimental results show that our proposed method outperforms traditional numerical optimization methods in surface optimization, especially for complex surfaces, and reduces the optimization time by multiple orders of magnitude.

Acknowledgments

This work was supported by National Key Research and Development Program of China under Grant (No.2022YFB2502903), National Natural Science Foundation of China under Grant (No.62088102), Basic research expenses of Xi'an Jiaotong University(No.xzy022023109).

References

Aumann, G. 2003. A simple algorithm for designing developable Bézier surfaces. *Computer Aided Geometric Design*, 20(8-9): 601–619.

Gu, X.; Gortler, S. J.; and Hoppe, H. 2002. Geometry images. In *Proceedings of the 29th annual conference on Computer graphics and interactive techniques*, 355–361.

Hanocka, R.; Hertz, A.; Fish, N.; Giryes, R.; Fleishman, S.; and Cohen-Or, D. 2019. Meshcnn: a network with an edge. *ACM Transactions on Graphics (ToG)*, 38(4): 1–12.

Hu, S.-M.; Liu, Z.-N.; Guo, M.-H.; Cai, J.-X.; Huang, J.; Mu, T.-J.; and Martin, R. R. 2022. Subdivision-based mesh convolution networks. *ACM Transactions on Graphics (TOG)*, 41(3): 1–16.

Hui, Z.; Gao, X.; Yang, Y.; and Wang, X. 2019. Lightweight image super-resolution with information multi-distillation network. In *Proceedings of the 27th acm international conference on multimedia*, 2024–2032.

Jiang, C.; Wang, C.; Rist, F.; Wallner, J.; and Pottmann, H. 2020. Quad-mesh based isometric mappings and developable surfaces. *ACM Transactions on Graphics (TOG)*, 39(4): 128–1.

Jiang, C.; Wang, H.; Inza, V. C.; Dellinger, F.; Rist, F.; Wallner, J.; and Pottmann, H. 2021. Using isometries for computational design and fabrication. *ACM Transactions on Graphics (TOG)*, 40(4): 1–12.

Ma, W.; Liu, Z.; Kudyshev, Z. A.; Boltasseva, A.; Cai, W.; and Liu, Y. 2021. Deep learning for the design of photonic structures. *Nature Photonics*, 15(2): 77–90.

Meyer, M.; Desbrun, M.; Schröder, P.; and Barr, A. H. 2003. Discrete differential-geometry operators for triangulated 2-manifolds. In *Visualization and mathematics III*, 35–57. Springer.

Nocedal, J.; and Wright, S. J. 1999. *Numerical optimization*. Springer.

Qi, C. R.; Su, H.; Mo, K.; and Guibas, L. J. 2017. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 652–660.

Ren, J.; An, N.; Zhang, Y.; Wang, D.; Sun, Z.; Lin, C.; Cui, W.; Wang, W.; Zhou, Y.; Zhang, W.; et al. 2023. SUGAR: Spherical Ultrafast Graph Attention Framework for Cortical Surface Registration. *arXiv preprint arXiv:2307.00511*.

Shen, Y.; Fu, H.; Du, Z.; Chen, X.; Burnaev, E.; Zorin, D.; Zhou, K.; and Zheng, Y. 2022. GCN-denoiser: mesh denoising with graph convolutional networks. *ACM Transactions on Graphics (TOG)*, 41(1): 1–14.

Sinha, A.; Bai, J.; and Ramani, K. 2016. Deep learning 3D shape surfaces using geometry images. In *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part VI 14*, 223–240. Springer.

Wang, H.; Guo, J.; Yan, D.-M.; Quan, W.; and Zhang, X. 2018. Learning 3d keypoint descriptors for non-rigid shape matching. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 3–19.

Wang, P.-S.; Liu, Y.; and Tong, X. 2016. Mesh denoising via cascaded normal regression. *ACM Trans. Graph.*, 35(6): 232–1.

Yuan, C.; Cao, N.; and Shi, Y. 2023. A Survey of Developable Surfaces: From Shape Modeling to Manufacturing. *arXiv preprint arXiv:2304.09587*.

Zhang, D.; Lu, X.; Qin, H.; and He, Y. 2020. Pointfilter: Point cloud filtering via encoder-decoder modeling. *IEEE Transactions on Visualization and Computer Graphics*, 27(3): 2015–2027.

Zhang, K.; Zuo, W.; Chen, Y.; Meng, D.; and Zhang, L. 2017. Beyond a gaussian denoiser: Residual learning of deep cnn for image denoising. *IEEE transactions on image processing*, 26(7): 3142–3155.

Zhao, W.; Liu, X.; Jiang, J.; Zhao, D.; Li, G.; and Ji, X. 2022. Local Surface Descriptor for Geometry and Feature Preserved Mesh Denoising. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, 3446–3453.

Zhao, W.; Liu, X.; Zhao, Y.; Fan, X.; and Zhao, D. 2019. NormalNet: Learning-based normal filtering for mesh denoising. *arXiv preprint arXiv:1903.04015*.