Shape Optimization of Grid Shells via Graph Attention Neural Networks

Andrea Favilli[1,2,*], Francesco Laccone[1], Paolo Cignoni[1], Luigi Malomo[1] and Daniela Giorgi[1]

[1]Institute of Information Science and Technologies "A. Faedo" (ISTI), National Research Council of Italy (CNR), via G. Moruzzi 1, Pisa, 56124, Italy
[2]University of Pisa, Lungarno Pacinotti 43, Pisa, 56126, Italy

Abstract

We introduce a computational method for form-finding and shape optimization of triangle-tessellated grid shells, kinds of 3D frame structures to be used as roofs and façades. The design of shapes has become an easy task after the advent of software tools, however obtaining good structural performance is still difficult. Form-finding and shape optimization techniques are hence employed to enhance system efficiency and material savings. We perform shape alterations by improving the structural configuration, while the alterations are slight enough to preserve the original aesthetic intents. We employ an attention graph neural network model that learns optimal displacements of the nodal coordinates with a target goal of reducing a statics-based loss, and it manages complex shapes and irregular tessellations successfully. The learning model is assessed by discussing the sensitivity of results towards the input features. Then, a variety of examples is provided to prove the effectiveness of our method, even in comparison with some classical form-finding tools which are not based on neural networks. The results show how our method overcomes the classical form-finding tools in terms of geometry preservation, on both the inner surface and the boundary.

Keywords

Grid Shells, Form-finding, Deep Learning, Graph Attention Networks, Structural Design

1. Introduction

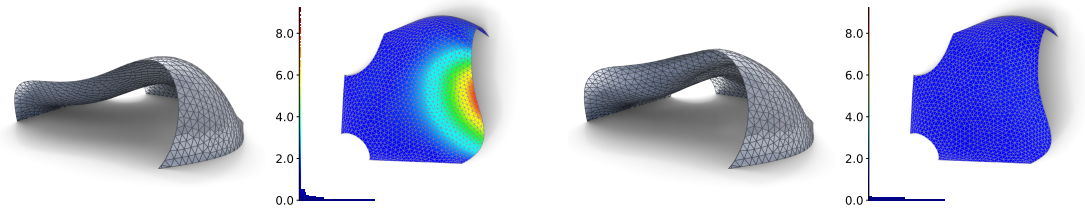
Developing methods to assess the architectural feasibility of free-form surfaces is a strong necessity, considered the increasing power of the available computational means to support shape modeling. If surfaces are more and more complicated on one side, fabrication issues pose increasingly hard challenges on the other side. Indeed, a shape is made buildable only when appropriate structural constraints are satisfied: statics, lighting, space requirements, costs, economy of materials and sustainability.

Geometry, which has a crucial role in finding aesthetically pleasing surfaces, is also the key to fix many feasibility issues. The discipline of *architectural geometry* [1] has emerged to give prominence to geometric solutions of fabrication problems. Contributions in the field are various, from paneling to equilibrium problems, and a broad amount of knowledge from mathematics and computer science is involved. The state of the art features a wide employment of discrete differential geometry, numerical analysis and optimization [2], while artificial intelligence and deep learning found less space to date, especially for what concerns 3D data. Considering the number of improvements the artificial intelligence made in several and different contexts, the expectation on results is high even in this field.

In this work, we leverage on graph attention neural network models to perform statics-driven shape modification on grid shells (Figure 1). Grid shells are architectural mesh structures made of beams which are jointed at the nodes, the beams support the whole amount of the structural load. The neural network inputs an original grid shell concept and decides the optimal shape by providing translation vectors yielding nodal displacements. The shape modification ensures a better inner force configuration and consequently a reduction of the global strain energy, while the boundary prescriptions and the visual aspect of the original concept are preserved.

Common machine learning approaches usually require a huge volume of pre-solved examples in order to generalize the solution of the task on new data. However, solving a single shape modification instance manually takes time and the necessity of finding the suitable expertise, with the non-uniqueness of ground truth in terms of design preservation. A problem which is hard to solve and a field, the architecture, which lacks on the public availability of digitized 3D shapes drove us to follow a dataset-free approach. The neural network learns the optimal nodal displacements for a given input shape after an iterative procedure based on the shape itself, the mean strain energy of the grid shell is expressed as a differentiable function from which we backpropagate to update the network weights. The backpropagation is performed through the automatic differentiation and the gradient

Ital-IA 2023: 3rd National Conference on Artificial Intelligence, organized by CINI, May 29–31, 2023, Pisa, Italy
*Corresponding author.
andrea.favilli@isti.cnr.it (A. Favilli)
0009-0004-9342-7106 (A. Favilli); 0000-0002-3787-7215 (F. Laccone); 0000-0002-2686-8567 (P. Cignoni); 0000-0001-7892-894X (L. Malomo); 0000-0002-6752-6918 (D. Giorgi)



(a) (b) (c) (d)

Figure 1: (a) We feed our 3D deep learning tool with an input grid shell free-form, provided as a triangular mesh. (b) The deformation the input would undergo as a consequence of Service Load. Euclidean norm of deformation is color mapped and expressed in meters with frequency histograms per node. (c) The output optimized grid-shell, preserving the original aesthetic while deformation is strongly reduced (d).

descent procedure is stopped after that the loss converges to a minimum.

Automatic differentiation (AD) is a way to evaluate derivatives which is often used in neural network training. It is preciser than numeric finite differences and faster than symbolic computations. AD keeps track of the computational graph of a given query function and it provides the evaluation of the derivative at a given point by visiting the computational graph backwards. The answer is assembled through the differentiation chain rule, knowing how to compute the derivatives of all the elementary operations involved. Beyond machine learning, AD is also employed in different architectural design problems: topology optimization [3], optimization of shells through NURBS (Non-Uniform Rational B-Splines) control structures [4], and in the context of CEM (Combinatorial Equilibrium Model) [5]. Neverthless, these contributions take into account simpler structures with less degrees of freedom, never employing full neural network models [6].

Our shape modification task fits the context of two important problems in architecture: *form-finding* and *shape optimization*. Form-finding aims at static optimality starting from the shape assumed by hanging membranes under the action of load. This approach, which is historically employed by famous designers like Gaudi and Isler, is overcomed by shape optmization in modern research. Instead of starting from a membrane, shape optimization leverages on genetic algorithms or gradient-based approaches to give static-awareness to reference shapes provided as input. Widely used software tools are based on both form-finding [7, 8] and shape optimization [9, 10, 11].

Our tool relies on the extensive knowledge from *geometric deep learning* [12]: convolution notions from machine learning on images are extended on more complex and structured input data like graphs and three-dimensional objects. While the convolution on 2D aggregates information via kernels on the neighboring pixels, switching on 3D means to rethink which the equivalent of pixels is, and how neighbors are defined. Dataset-free approaches and single instance learning are not new in literature [13], even if this paradigm is less considered.

In Section 2 we develop some more details about the method, while in Section 3 we report the settings of the experiments and we provide some results, showing how our tool meets the expectations both in terms of static-awareness and design preservation.

2. Methods

Figure 2 shows how our method works. The most natural data representation for a triangle-tessellated grid shell is a mesh $\mathscr{M} = (V, E, F)$, where V are the vertices, $E \subseteq V \times V$ are the edges and $F \subseteq V \times V \times V$ are the faces. The structural nodes of the grid shell are identified with the mesh vertices and the beams are identified with the mesh edges.

We consider mean strain energy

$$\mathscr{L}(\mathscr{M}) = \frac{1}{|E|} \sum_{e \in E} S_e, \tag{1}$$

where S_e are per edge energies. More information about S_e definition and computation can be found on Section 2.3. If T_θ is the neural model depending on a set of learnable parameters $\boldsymbol{\theta}$, our shape modification task can be formalized as follows: we search for an optimal mesh

$$\mathscr{M}^* := T_{\boldsymbol{\theta}}(\mathscr{M})$$

such that

$$\boldsymbol{\theta}^* \in \operatorname{argmin}_{\boldsymbol{\theta}} \mathscr{L}(T_{\boldsymbol{\theta}}(M)). \tag{2}$$

A subset of the boundary vertices $\overline{\partial V} \subseteq \partial V$, usually the ones placed on the ground, is kept fixed while the model is free to displace the remainder vertices in $\partial V \setminus \overline{\partial V}$

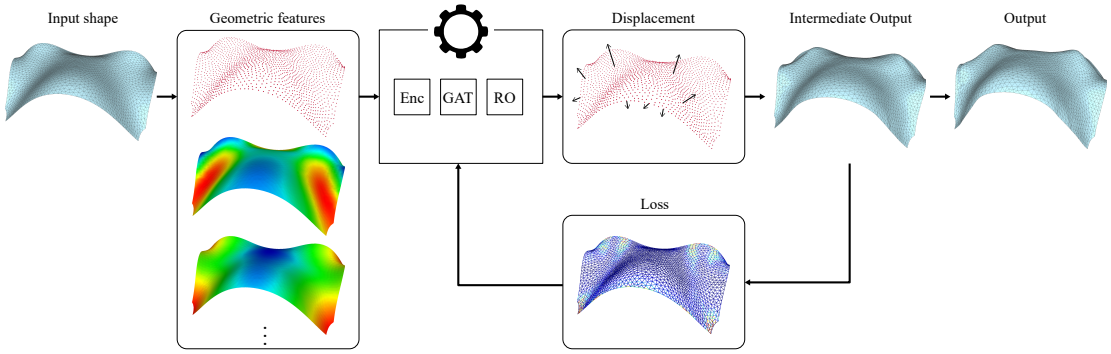


Figure 2: The pipeline of the tool. The input 3D mesh is encoded through a set of intrinsic and extrinsic geometric input features (vertex coordinates, normals, curvature, geodesic distances and centrality measures) and fed to the 3D deep learning network. The network is made of three modules: a feature encoder (Enc), four attention layers (GAT) and a readout function (RO) which outputs the nodal displacements. Network weights are updated according to gradients of a loss function based on edge mean strain energy, updates are repeated until the loss converges and the last iteration output is the inferred shape.

2.1. Input features

We select three clusters of input features for each vertex $\mathbf{v} \in V$ (base, geometric and geodesic) so that each vertex is fed to the network through an input feature vector $\mathbf{x}_\mathbf{v} \in \mathbb{R}^{12}$. Input features are selected among the quantities shown to have relevance in the architectural geometry context.

Base features are vertex coordinates $\mathbf{v} \in \mathbb{R}^3$ and vertex normals $\mathbf{n}(\mathbf{v}) \in \mathbb{R}^3$. These features are referred as base because are traditionally employed in 3D learning applications, [14].

Curvature features are the principal curvatures $\kappa_1(\mathbf{v})$, $\kappa_2(\mathbf{v})$. Curvatures and their variation are related to the distribution of the structure inner forces. Indeed, when the external load can be commuted into planar inner forces structural efficiency is reached.

Geodesic features relate local information to global information giving the relative position of vertices in the whole shape. We consider the distance $d(\mathbf{v}, \partial V)$ of vertices from the set ∂V of boundary vertices and the distance $d(\mathbf{v}, \overline{\partial V})$ from the set $\overline{\partial V}$ of fixed boundary vertices. We recall that

$$d(\mathbf{v}, X) := \min\{d(\mathbf{v}, \mathbf{w}) \mid \mathbf{w} \in X\},$$

where $d(\mathbf{v}, \mathbf{w})$ is the geodesic distance between two mesh vertices, i.e. the length of the shortest edge path connecting them. We also consider centrality measures $\mathscr{C}_{\partial V}(\mathbf{v})$ and $\mathscr{C}_{\overline{\partial V}}(\mathbf{v})$, where the centrality of a vertex $\mathbf{v}$ to a set X is

$$\mathscr{C}_X(\mathbf{v}) := \sum_{\mathbf{w} \in X} d(\mathbf{v}, \mathbf{w}).$$

Each feature cluster (having dimension n_i, $i = 1, 2, 3$) is mapped to 256 channels through the encoding

$$\mathbf{h}_{i,\boldsymbol{\theta}}(\mathbf{x}) := \tanh(W_{i,\boldsymbol{\theta}}\mathbf{x} + \mathbf{b}_{,\boldsymbol{\theta}})$$

with $W_{i,\boldsymbol{\theta}} \in \mathbb{R}^{256 \times n_i}$ and $\mathbf{b}_i^{\boldsymbol{\theta}} \in \mathbb{R}^{256}$; the activation is hyperbolic tangent in order to bind mapped features in the interval $(-1, 1)$.

The input feature vector $\mathbf{x}_\mathbf{v} \in \mathbb{R}^{12}$ is encoded to $\mathbf{x}_\mathbf{v}^{enc} \in \mathbb{R}^{768}$, this last is then fed to a sequence of four GATv2 [15] attention layers. GATv2 layers process the k-nearest neighbor graphs built on the vertex feature spaces with the Euclidean metric.

2.2. Learning model

The deep learning model architecture is shown in Figure 3. In the previous Section we showed how the input features are selected and encoded, we now consider the GATv2 attention layers. We denote $\mathbf{x}_\mathbf{v}^\ell$ the input features of the layer ℓ for $\ell = 1, \dots, 4$, $\mathbf{x}_\mathbf{v}^1 = \mathbf{x}_\mathbf{v}^{enc}$ by construction. Each layer has its own nearest neighbor graph structure induced by the Euclidean distance on the corresponding feature space: we denote with $\mathcal{N}_\mathbf{v}$ the set of all the neighbors of the vertex $\mathbf{v} \in V$. GATv2 layers transform feature vectors according to learned weighted means on the neighborhoods, namely

$$\mathbf{x}_\mathbf{v}^{\ell+1} := \alpha_{\mathbf{vv}} W_{\boldsymbol{\theta}} \mathbf{x}_\mathbf{v}^\ell + \sum_{\mathbf{w} \in \mathcal{N}_\mathbf{v}} \alpha_{\mathbf{vw}} W_{\boldsymbol{\theta}} \mathbf{x}_\mathbf{w}^\ell. \tag{3}$$

$W_{\boldsymbol{\theta}}$ and $\mathbf{a}_{\boldsymbol{\theta}}$ are arrays of learnable parameters which allow to compute the weights

$$\alpha_{\mathbf{vw}} := \frac{\exp\left(\mathbf{a}_{\boldsymbol{\theta}}^T \phi(W_{\boldsymbol{\theta}}(\mathbf{x}_\mathbf{v} \,\|\, \mathbf{x}_\mathbf{w}))\right)}{\sum_{\mathbf{w} \in \mathcal{N}_\mathbf{v} \cup \{\mathbf{v}\}} \exp\left(\mathbf{a}_{\boldsymbol{\theta}}^T \phi(W_{\boldsymbol{\theta}}(\mathbf{x}_\mathbf{v} \,\|\, \mathbf{x}_\mathbf{w}))\right)} \tag{4}$$

and ϕ is a *leaky ReLU* activation function.

We chain the four 256-dimensional GATv2 outputs to produce a deep feature vector $\widetilde{\mathbf{x}_\mathbf{v}}$ and the final displacement for the vertex $\mathbf{v}$ is $\delta_\mathbf{v} := \mathbf{k}_{\boldsymbol{\theta}}(\widetilde{\mathbf{x}_\mathbf{v}})$, where $\mathbf{k}_{\boldsymbol{\theta}}$ is the

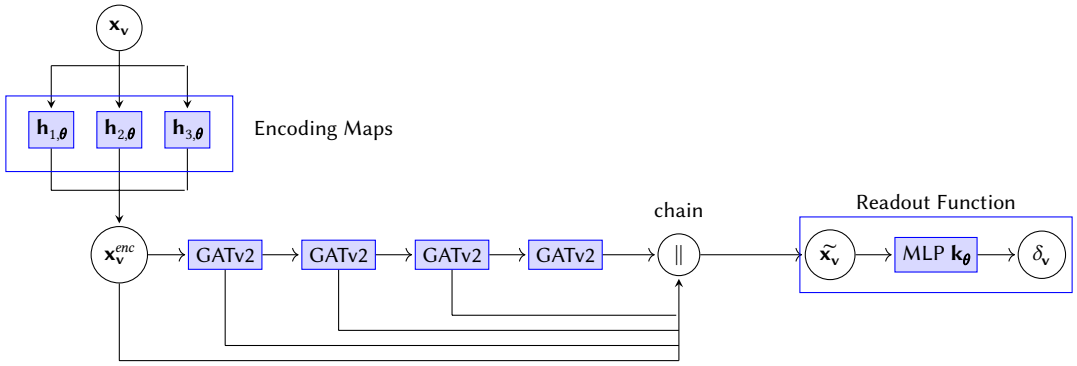


Figure 3: Here is how the learning model works. The feature vector $\mathbf{x}_v$ is encoded in $\mathbf{x}_v^{enc}$, which is subsequently fed to four GATv2 layers to get a deep vector $\widetilde{\mathbf{x}}_v$ after layer output concatenation. Finally, a multilayer perceptron plays the role of a readout map to produce vertex displacements δ_v from the deep vectors.

readout function implemented as a multilayer perceptron, shared on the whole mesh.

2.3. Loss and boundary constraints

We perform gradient-based optimization on the loss function

$$\mathscr{L} + \zeta \mathscr{B},$$

where $\mathscr{L}$ is the mean strain energy defined on Equation 1, $\mathscr{B}$ is a penalty term to ensure that the boundary fixed vertices in $\overline{\partial V}$ have no displacement, and ζ is a weighting factor. ζ is set at the first iteration according to the values of $\mathscr{L}$ and $\mathscr{B}$ so that $\zeta\mathscr{B}$ is the 30% of $\mathscr{L}$. A universal value for ζ cannot be found due to the order of magnitude of $\mathscr{L}$ that changes with the scale of the grid shell structure.

The implementation for $\mathscr{L}$ is written in PyTorch [16]. We adopt a smart, tensor-based approach to compute all the S_e from Equation 1 in parallel, to make the computation profitable to be run on GPUs. If we consider beam local frames $\{\mathbf{x}_e, \mathbf{y}_e, \mathbf{z}_e\}$, each beam is subjected to six stress components: axial elongation and torsion along $\mathbf{x}_e$, and transverse bending and shear along planes $\{\mathbf{x}_e, \mathbf{y}_e\}$ and $\{\mathbf{x}_e, \mathbf{z}_e\}$ [17]. The beam energy S_e is a sum over the six components contributions, and it is computed after a coordinate change to the observer's reference system and back.

The penalty term $\mathscr{B}$ is just the sum of the Euclidean norms of the displacements outputted by the network model for the vertices in $\overline{\partial V}$, namely

$$\mathscr{B} := \frac{1}{|\overline{\partial V}|} \sum_{\mathbf{v} \in \overline{V}} \|\delta_\mathbf{v}\|. \tag{5}$$

After the last training iteration, at inference time, $\mathscr{B}$ is so small that the displacements for $\overline{\partial V}$ can be set to zero without consequences to the shape and the static analysis.

3. Settings and Results

In this section we briefly summarize the considerations we made after the assessment of our learning model on fifteen different free-forms. Some examples and the most relevant considerations are reported, focusing on parameter settings, iteration times, sensitivity to the geometric input features and comparisons with other form-finding tools.

Gradient based optimization is performed via Stochastic Gradient Descent (SGD), momentum is set to 0.9, learning rate is changed according to the scale of the input shell. More precisely, we consider structures with beam cross section radius of 0.017 m at a learning rate of 0.01 and structures with cross section radius of 0.08 m at a learning rate of 0.0005. Loading of 3 kN/m^2 is vertical, in the gravity direction, and is distributed on the nodes by tributary area. The weight of the beams is then included as a extra lumped load, assuming the material density as $78.5kN/m^3$.

We test our method on a Microsoft Windows® 10 Pro machine with a i7-6700K CPU, 32 GB of RAM, a NVIDIA GeForce GTX 1080 GPU with 8 GB of dedicated memory. Loss and iteration times are reasonable, and the advantage of loss computing on GPUs rather than CPUs is noticeable. For example, in the instance of Figure 4 mean differentiation time over iterations is 0.128 seconds on GPU against 1.02 seconds on CPU.

3.1. Role of features and attention

We discovered the relevance of feature clusters and attentions by observing how the output changes if we remove the given cluster or we change the Graph Neural Network paradigm. More in detail, we assess the role of attention mechanisms by comparing GATv2 with the plain graph edge convolution (DGCNN, [18]).

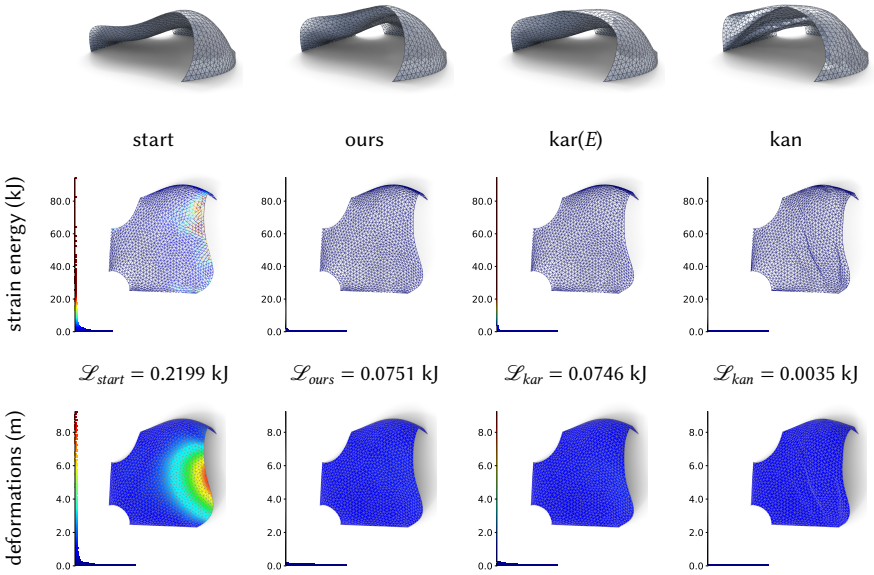


Figure 4: Comparison of results from our model and form-finding tools for similar strain energy.

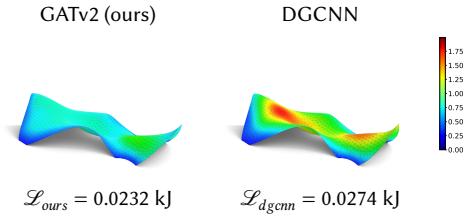


Figure 5: The results with attention layers (middle) and with simple dynamic edge convolution (left). Color mapping refer to the vertex displacements of the model outputs from the input. Attention layers offer better shape preservation with similar static performance. Models: Gopher, learning rate 0.1, 200 SGD iterations; Wave, learning rate 0.01, 200 SGD iterations.

For what concerns geometric features, the use of curvatures is related to a reduced Hausdorff distance between the input shape and the output, for similar strain energy configurations. This is a good indicator of a lesser shape modification which is preferable in terms of shape design preservation intents.

Geodesic features are instead related to a greater shape fairness, together with a faster convergence of the method and smoother loss descent curves.

The advantage of attention mechanism over the edge convolution is on the lesser triviality of the produced shapes. The outputs from the DGCNN cases are often pulled-up version of the inputs (Figure 5). If adding material reduces strain energy on one side it increases the norms of displacements on the other side, not preserving hence the original design intent.

3.2. Comparison with other form-finding tools

We compare our model outputs with two popular form-finding tools, Karamba [19] and Kangaroo [20]. These two approaches are different in terms of both the method and the parameters the user can control. In order to do a fair comparison with our model we generate some form-finding instances and we select the ones with similar energy configuration with respect to ours. A full overview is shown in Figure 4.

Our model better performs in terms of shape preservation, both the form-finding tools alter free boundaries (made of $\overline{\partial V} \setminus \partial V$ vertices) and design traits like creases or spikes. This phenomenon, which is noticeable in Karamba, is even bigger with Kangaroo.

4. Conclusions

We develop a 3D geometric deep learning method which performs well on both form-finding and shape optimization tasks. The advantages with the other state-of-the-art tools are a lesser number of tunable parameters and an enhanced performance in terms of design preservation.

Possible future work is in the direction of learning rate adaptivity, not only with respect to the input model scale but also with respect some design-related user requests. Morevorer, suitable topological descriptors [21] can be exploited as sophisticated regularizers to achieve even finer results in terms of shape preservation.

References

[1] H. Pottmann, D. Bentley, A. Asperl, M. Hofer, A. Kilian, Architectural Geometry, number v. 10 in Architectural geometry, Bentley Institute Press, 2007. URL: https://books.google.it/books?id=bIceAQAAIAAJ.

[2] H. Pottmann, M. Eigensatz, A. Vaxman, J. Wallner, Architectural geometry, volume 47, 2015, pp. 145–164. doi:10.1016/j.cag.2014.11.002.

[3] S. Norgaard, M. Sagebaum, N. Gauger, B. S. Lazarov, Applications of automatic differentiation in topology optimization, volume 56, 2017, p. 1135–1146. doi:https://doi.org/10.1007/s00158-017-1708-2.

[4] B. Yang, J.-L. Ma, Q. Zhang, Shape optimization of shell structures based on nurbs description using genetic algorithm, 2013. doi:10.13140/RG.2.1.3188.5525.

[5] R. Pastrana, P. O. Ohlbrock, T. Oberbichler, P. D'Acunto, S. Parascho, Constrained form-finding of tension–compression structures using automatic differentiation, Computer-Aided Design 155 (2023) 103435. URL: https://www.sciencedirect.com/science/article/pii/S0010448522001683. doi:https://doi.org/10.1016/j.cad.2022.103435.

[6] G. Wu, A framework for structural shape optimization based on automatic differentiation, the adjoint method and accelerated linear algebra, 2022. arXiv:2211.15409.

[7] K.-U. Bletzinger, E. Ramm, Structural optimization and form finding of light weight structures, volume 79, Elsevier, 2001, pp. 2053–2062. doi:10.1016/S0045-7949(01)00052-9.

[8] D. Veenendaal, P. Block, An overview and comparison of structural form finding methods for general networks, volume 49, 2012, pp. 3741–3753. doi:10.1016/j.ijsolstr.2012.08.008.

[9] S. Fujita, M. Ohsaki, Shape optimization of free-form shells using invariants of parametric surface, volume 25, SAGE Publications Sage UK: London, England, 2010, pp. 143–157. doi:10.1260/0266-3511.25.3.143.

[10] M. Firl, R. Wüchner, K.-U. Bletzinger, Regularization of shape optimization problems using fe-based parametrization, volume 47, Springer-Verlag, Berlin, Heidelberg, 2013, p. 507–521. doi:10.1007/s00158-012-0843-z.

[11] H. Wang, Z. Chen, G. Wen, G. Ji, Y. M. Xie, A robust node-shifting method for shape optimization of irregular gridshell structures, in: Structures, volume 34, Elsevier, 2021, pp. 666–677. doi:10.1016/j.istruc.2021.08.003.

[12] M. M. Bronstein, J. Bruna, T. Cohen, P. Velickovic, Geometric deep learning: Grids, groups, graphs, geodesics, and gauges, CoRR abs/2104.13478 (2021). URL: https://arxiv.org/abs/2104.13478. arXiv:2104.13478.

[13] R. Hanocka, G. Metzer, R. Giryes, D. Cohen-Or, Point2mesh: A self-prior for deformable meshes, volume 39, Association for Computing Machinery, New York, NY, USA, 2020. URL: https://doi.org/10.1145/3386569.3392415. doi:10.1145/3386569.3392415.

[14] C. R. Qi, L. Yi, H. Su, L. J. Guibas, Pointnet++: Deep hierarchical feature learning on point sets in a metric space, in: Proceedings of the 31st International Conference on Neural Information Processing Systems, NIPS'17, Curran Associates Inc., Red Hook, NY, USA, 2017, p. 5105–5114.

[15] S. Brody, U. Alon, E. Yahav, How attentive are graph attention networks?, in: International Conference on Learning Representations, 2022. URL: https://openreview.net/forum?id=F72ximsx7C1.

[16] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, A. Lerer, Automatic differentiation in pytorch, in: NIPS 2017 Workshop on Autodiff, 2017. URL: https://openreview.net/forum?id=BJJsrmfCZ.

[17] J. F. Doyle, Static and Dynamic Analysis of Structures: with An Emphasis on Mechanics and Computer Matrix Methods, Solid Mechanics and Its Applications №6, 1 ed., Springer Netherlands, 1991. doi:https://doi.org/10.1007/978-94-011-3420-0.

[18] Y. Wang, Y. Sun, Z. Liu, S. E. Sarma, M. M. Bronstein, J. M. Solomon, Dynamic graph cnn for learning on point clouds, volume 38, Association for Computing Machinery, New York, NY, USA, 2019. URL: https://doi.org/10.1145/3326362. doi:10.1145/3326362.

[19] C. Preisinger, M. Heimrath, Karamba - A toolkit for parametric structural design, volume 24, Taylor & Francis, 2014, pp. 217–221. doi:10.2749/101686614X13830790993483.

[20] D. Piker, Kangaroo: form finding with computational physics, volume 83, Wiley Online Library, 2013, pp. 136–137. doi:10.1002/ad.1569.

[21] F. Hensel, M. Moor, B. Rieck, A survey of topological machine learning methods, Frontiers in Artificial Intelligence 4 (2021). URL: https://www.frontiersin.org/articles/10.3389/frai.2021.681108. doi:10.3389/frai.2021.681108.