ISSN 2464-4617 (print)
ISSN 2464-4625 (online)
Computer Science Research Notes - CSRN
http://www.wscg.eu
WSCG 2025 Proceedings

Learning 2D Triangular Meshes from Images with Transformers

Maximilian Zöch
Fraunhofer Austria Research GmbH
Graz University of Technology, Institute for Visual Computing
Austria, Graz
maxzoech@icloud.com

Ulrich Krispel
Fraunhofer Austria Research GmbH
Graz University of Technology, Institute for Visual Computing
Austria, Graz
ulrich.krispel@fraunhofer.at

Ursula Augsdörfer
Graz University of Technology, Institute for Visual Computing
Austria, Graz
augsdoerfer@tugraz.at

Abstract

In this work, we present a method for learning 2D triangular meshes from images using the Transformer architecture. Creating polygonal representations of 2D surfaces from images has a wide range of applications, e.g. texture mapping and image segmentation. Current machine learning based methods either rely on deforming template meshes with fixed topology or require expensive post-processing to extract a planar polygonal mesh. In this paper, we demonstrate that deep learning can be utilized to directly generate a planar polygonal surface from arbitrary images without the need for additional input or constraints. Specifically, we show that the attention mechanism in transformer networks is highly effective in learning a unified representation of vertex positions and their neighborhood relationships. To showcase this, we propose a self-supervised training pipeline that enables end-to-end learning of meshes directly from images. We apply our approach to various datasets of handwritten letters and digits, and show that the model is capable of learning meshes with varying genus.

Keywords

polygon mesh generation, machine learning, transformer networks

1 INTRODUCTION

Digital representations of real-world objects are ubiquitous in todays world with applications in both industrial engineering and entertainment. Learning representative and robust features for reconstructing shape from measurements is therefore an important task, and gains momentum due to the advancements in sensing devices and the increasing amount of available data. Such a representation enables conditional generation of specific objects, which has been applied with great success in the image domain – using e.g. Generative Adversarial Networks (GANs) [Goo+14; ACB17; Gul+17; Kar+18; BDS18] or diffusion models [SD+15; SE19; HJA20; Rom+22; Nic+22; Sah+22] to generate images. These methods are inherently tied to the geometric representation – regular grids in the case of images. To learn irregular structures, various geometric representations are common, especially in the 3D domain: point clouds [FSG17], implicit functions [Par+19; Mil+20] and meshes [Wan+18]. While meshes are the prominent representation for rendering and shape processing, not many works are concerned with direct learning of such irregular structures.

In contrast to existing work, which consists of multi-stage procedures with conversions between geometry representations [GMJ19], we present a novel approach towards direct regression of irregular geometric representations: We train a non-autoregressive Transformer model to learn to predict the surface of an input shape as triangular mesh (see Figure 1) from an input image.

Our novel decoder structure directly predicts vertex positions and a surface matrix, which is similar to an adjacency matrix, and derives triangle visibility. Unlike existing work on generating meshes using autoregressive transformers to sequentially generate geometry [Nas+20; Sid+23; Hao+24], we utilize the permutation invariant structure of transformers directly. Our approach only requires a single model evaluation during inference to generate 2D meshes from an input image. Although the number of vertices is fixed for a network, the genus and thus the number of generated triangle faces vary to provide the most suitable solution for a given input image.

The paper is structured as follows: In Section 2, we give an overview over geometry representations and respective deep learning methods. The approach on how to build the training pipeline is described in detail in Section 3; We quantitatively evaluate mesh quality and qualitatively analyze the generalization capability in Section 4, and conclude with Section 5.

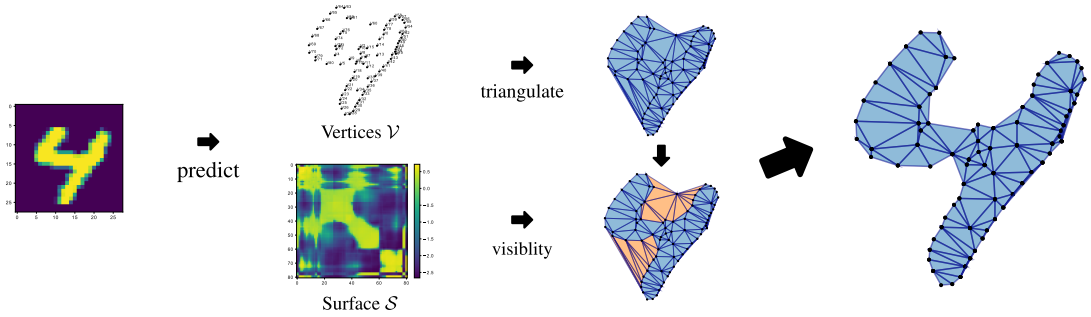


Figure 1: Our transformer model learns to predict a triangulation given an input image, by predicting a set of vertices $\mathcal{V}$ that are triangulated via Delaunay triangulation. The model furthermore predicts a surface matrix $\mathcal{S}$ containing edge weights. From this matrix, triangle visibility is inferred yielding output shapes of arbitrary topology. The approach is formulated a self-supervised end-to-end manner using differentiable triangulation and rasterization.

2 RELATED WORK

The generation of 2D meshes ("meshing") is a long-standing problem with many written works on the topic [Fen+18; She12]. These methods typically utilize specific algorithmic approaches, e.g. spatial search structures, such as quad-trees, or Delaunay methods. In contrast, we demonstrate learning the shape representation in a fully data driven way using machine learning. Our method is built on the transformer architecture. We first give an overview of different geometry representations that have been proposed for learning shape representations with transformer. Then, we elaborate on important components on which our approach is built.

2.1 Geometry Representation

Regular Structures

Early work [Wu+15; Wu+16; Cho+16] in 3D transformer represented shapes as voxels. Voxels generalize image pixels to the 3D domain, which makes it easy to adapt models originally devised for images. For example, Wu et al. [Wu+16] implemented a GAN [Goo+14] that uses 3D CNNs to generate shapes. One major drawback of voxel representations is that they need to store a regularly sampled dense volume, which makes them costly to scale.

Irregular Structures

Meshes represent shapes as a set of points and explicit surface information. Pixel2Mesh [Wan+18] and follow up work [Wen+19] reconstruct a 3D mesh from RGB images by deforming a template mesh. Later work extended this approach by modifying the topology of a template mesh to generate meshes with varying genus [Pan+19; BC+22]. Another line of research is learning multiple disjunct patches that represent a surface [Gro+18; BH+18]. PolyGen [Nas+20] uses an autoregressive transformer to generate vertices, and then connect them to n-gon surfaces using a PointerNet [VFJ15].

2.2 Transformers

Transformers are a popular class of transformer models first introduced in [Vas+17] for machine translation. Typically, input and output are represented by a sequence of tokens which are transformed by the model. Transformers are not sequential models, therefore, each token requires an additional positional embedding. The architecture has been successfully applied to a wide range of NLP and computer vision tasks. The core of a transformer model is the scaled dot product attention which computes a similarity score between the linearly transformed input tokens Q, K and V and is computed as:

$$\text{Attention}(Q,K,V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

The scaling factor of $1/\sqrt{d_k}$ provides a normalization of the dot product with query and key dimension d_k. In the case of self attention, query (Q), key (K), and value (V) are all linear transformations of the latent space. This allows the architecture to model self-similarity. When performing cross attention, the query is a linear transformation of encoded tokens from a different sequence, while key and value are the same as for self-attention. Cross attention is used to model the similarity between two different sequences.

Transformers can be broadly categorized as encoder-only, decoder-only, and encoder-decoder models. Encoder-only models such as BERT [Dev+19] or vision transformer (ViT) [Dos+20] learn a representation over a set of input tokens. Decoder-only models like the GPT family of models [Rad+19; Bro+20] generate tokens in an autoregressive manner, starting from an input token. Encoder-decoder models use an encoder to learn a representation from the input sequence to condition the decoder [Vas+17; Lew+20]. In this paper we implement an encoder-decoder style model, but use a non-autoregressive decoder.

Thus, unlike other transformer based mesh generation frameworks [Nas+20], our model does not need to be evaluated multiple times at inference time to sequentially generate an output mesh.

2.3 Differentiable Graphics Primitives

During training, differential graphics primitives are used to formulate an end-to-end pipeline, rasterizing the predicted mesh.

Differentiable Rendering Rendering is the process of producing an image from a scene that is composed of geometry, lights, and cameras. Differentiable rendering aims to define the gradient of a pixel with respect to these scene parameters. One can then formulate an optimization problem to recover e.g. camera pose, geometry or materials from a rendered image. *SoftRasterizer* [Liu+19] approximates the rendering with a probabilistic approach: Probability maps of triangles based on the signed screen space distance are used to formulate rasterization in a differentiable manner. For each pixel, all probability maps are aggregated. As calculating the distances between all pixels and all triangles would be prohibitively expensive, only a local neighborhood of k-nearest triangles is considered for gradient propagation. A major limitation of this approach is that it does only support direct illumination. Later works aimed to implement physically-based differentiable path tracing for more photorealistic results [Li+18; LHJ19; Zha+21], however, the SoftRasterizer is sufficient for our approach.

In our approach we will make use of differential rendering to optimize the structure of the predicted planar mesh during training.

Differentiable Surface Triangulation Triangle mesh optimization and surface remeshing are very well-studied problems in computational geometry [All+08; Kha+22; WLH16]. Recent work proposed a soft relaxation of the classical Delaunay triangulation [Rak+21], that jointly optimizes vertex positions and triangle mesh connectivity. It defines a differentiable continuous triangle inclusion score based on the sigmoid of the signed distance of the triangle circumcenter to the so-called reduced Voronoi cells of the triangle vertices. Voronoi cells are built over only k-nearest neighbors of a vertex to account for the cubic growth of all possible triangles. The authors construct a low distortion parametrization of a 3D surface into 2D patches which are differentiable using the aforementioned method, and show results for two different optimization criteria: triangle size and vector-field alignment.

Our approach utilizes the inclusion score of this approach in our setup to propagate gradients only to triangles likely to satisfy the Delaunay criterion.

3 METHOD

The main goal of this paper is to demonstrate that the transformer architecture is an effective tool for learning a latent space representation that simultaneously predicts vertices *and* neighborhood information. We construct a triangle mesh via unconstrained Delaunay triangulation of the predicted vertices and use the learned neighborhood information to determine triangle visibility; the Delaunay triangulation guides the model towards desirable geometry. During training, meshes are rendered using a differentiable renderer, and the loss between the rendered image and target image is minimized. This means that the procedure does not require any ground-truth mesh data. We will now describe the mesh representation, model architecture, and training formulation in greater detail.

3.1 Mesh Representation

The aim is to generate the mesh $\mathscr{M} = (\mathscr{V}, \mathscr{F})$ of a shape consisting of vertices $\mathscr{V} = \{v_0, \ldots, v_n\}$ with $v_i \in \mathbb{R}^2$ and triangular faces $f_i \in \mathscr{F}$ with $f_i = \{v_j, v_k, v_l\}$ that approximate the surface of the shape. The model predicts the set of vertices $\mathscr{V}$ and a surface matrix $\mathscr{S} = \mathscr{V} \times \mathscr{V}$ that contains a weight entry per edge. This matrix and a Delaunay criterion is used to obtain $\mathscr{F}$ from the set of all possible triangles $\mathscr{F}'$, by first evaluation the delaunay triangulation $\mathscr{F}_{\text{triag}}$ of $\mathscr{V}$ and obtain triangle visibility $\mathscr{F}_{\text{surf}}$:

$$\mathscr{F}' = \mathscr{V} \times \mathscr{V} \times \mathscr{V} \tag{1}$$

$$\mathscr{F}_{\text{triag}} = \left\{ f_i \in \mathscr{F}' \mid f_i \text{ is delaunay} \right\} \tag{2}$$

$$\mathscr{F}_{\text{surf}} = \left\{ f_i \in \mathscr{F}' \mid s_i > 0 \right\} \tag{3}$$

$$\mathscr{F} = \mathscr{F}_{\text{triag}} \cap \mathscr{F}_{\text{surf}} \tag{4}$$

The surface score s_i^{surf} of face f_i is defined as the sum of the edge scores from the surface matrix $\mathscr{S}$:

$$s_i^{\text{surf}} = \mathscr{S}_{v_j,v_k} + \mathscr{S}_{v_k,v_l} + \mathscr{S}_{v_l,v_j} \tag{5}$$

In order to make the formulation end-to-end differentiable, we define a continuous face score s_i that indicates whether the face f_i is part of the mesh or not, We use the signed distance used for the continuous triangle inclusion score from Rakotosaona et al. [Rak+21] (see Section 2.3) as score s_i^{triag} to implement the Delaunay criterion:

$$s_i = \min\left(s_i^{\text{triag}}, s_i^{\text{surf}}\right) \tag{6}$$

The idea of the formulation in Equation 6 is that s_i^{triag} guides the model to learn desirable triangles, and s_i^{surf} conditions the latent representation to describe triangle visibility.

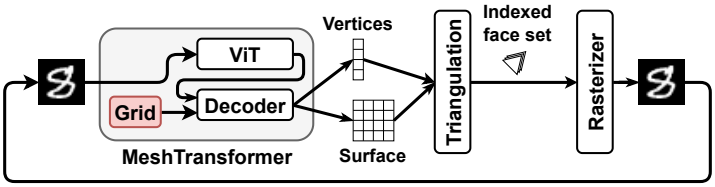


Figure 2: Training pipeline. Our model predicts a set of vertices $\mathscr{V}$ which are triangulated using differentiable triangulation and a surface matrix $\mathscr{S}$ that determines triangle visibility. Using differentiable rasterization, an output image is computed; the final loss is calculated between input and output image.

3.2 Training

During training we use a differentiable rasterizer called SoftRasterizer [Liu+19] to render the meshes. SoftRasterizer implements differentiable rasterization in two phases: Rasterization is relaxed as the probability of a pixel belonging to a triangle, which is implemented via the sigmoid σ of the signed screen distance d which is scaled by a constant factor $\gamma = 10^{-4}$ for the experiments in this paper. Afterwards, at each pixel position the probability maps are aggregated. We used the implementation of PyTorch3D [Rav+20] which only computes maps for the k-nearest faces for each pixel. We use this architecture to first raster probability maps for faces of $\mathscr{F}$ and perform weighted aggregation using its corresponding surface score; we implement the aggregation using alpha blending at each pixel position (xy) over the k-nearest faces.

$$\alpha_{k,(xy)} = \sigma\left(\frac{s_{k,(xy)}}{\gamma}\right) \cdot \sigma\left(\frac{-d_{k,(xy)}}{\gamma}\right) \tag{7}$$

$$\hat{I}_{(xy)} = 1 - \prod^{k} \left(1 - \alpha_{k,(xy)}\right) \tag{8}$$

We rasterize $\hat{I}$ at 56×56 and bilinearly upsample the training images I accordingly using bilinear interpolation. Finally, we calculate the silhouette intersection over union loss from *SoftRasterizer* as our training loss, $\otimes$ and $\oplus$ denote element-wise product and sum:

$$\mathscr{L} = \frac{||\hat{I} \otimes I||_1}{||\hat{I} \oplus I - \hat{I} \otimes I||_1} \tag{9}$$

3.3 Model Architecture

Our model architecture is closely modeled after the transformer first introduced by [Vas+17]; we use an encoder-decoder style architecture. The encoder learns a feature embedding from the input images. The decoder then deforms a set of input points, which are arranged in a grid, conditioned on the encoded features from the image.

Encoder Our encoder is a miniaturized version of the design proposed for the vision transformer (ViT) [Dos+20]. The input image is first split into 7×7 input patches of size 4×4 pixels which are then flattened passed to the transformer as tokens. Unlike for the decoder, the position of the patches in the image is important to capture the global shape, which is why we add a learned positional embedding to each of the tokens. We have experimented with different positional embedding schemes but found that they do not outperform learned embeddings and have thus chosen them for their simplicity.

Decoder Similar to [Yan+18] the decoder deforms a grid of 9×9 points. Like PointNet [Qi+17], the weights of the feed-forward neural network in the transformer are shared between all the points (tokens). The permutation invariant nature of the transformer allows us to represent the mesh as a set of points with neighborhood information. We thus do not add positional embeddings to the inputs of the decoder. Instead, we only pass the input points through a Fourier feature mapping [Tan+20], as neural networks have been shown to have a low frequency bias which makes it harder to learn fine structures [Rah+19].

The layers ("blocks") of the decoder follow standard transformer design: We use cross attention to condition the vertex position on the features learned by the encoder. Self attention is used to represent neighborhoods of vertices by making neighboring vertices more similar in the latent space. Thus, the latent space of the transformer jointly represents vertex positions and neighborhood information.

The final layer of the decoder is the geometry head that transforms the latent representation into the intermediate mesh $\mathscr{M} = (\mathscr{V}, \mathscr{S})$. In order to compute the surface matrix $\mathscr{S}$, we perform multi-head (symmetric) self-attention.

$$\mathscr{S} = \text{mlp}\left(\frac{QK^T}{\sqrt{d_k}}\right) \tag{10}$$

To guarantee the surface matrix to be symmetric, we share the weights of the linear transformations of key, query, and value in the self-attention layer. We then use a small multi-layer perceptron to merge the multiple surface matrices of the attention heads into a single one. We found that using a small neural network instead of a linear transformation significantly stabilizes training.

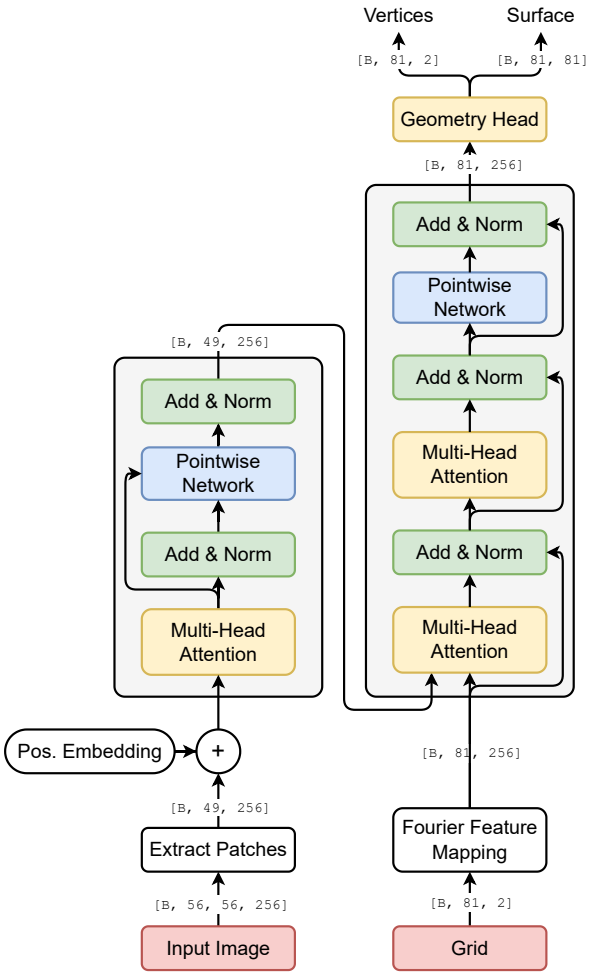


Figure 3: Model architecture. The encoder (left) encodes a sequence of image patches with batchsize B. The decoder (right) deforms a grid of 9×9 points according to the encoded image. The latent representation of the decoder is transformed into a vertex set $\mathscr{V}$ and surface matrix $\mathscr{S}$ by the geometry head.

The vertices $\mathscr{V}$ are computed by applying the self-attention matrix and then linearly mapping the vertices from latent space to $\mathbb{R}^2$.

$$\mathscr{V}_h = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{11}$$

$$\mathscr{V} = \text{linear}\left(\mathscr{V}_h\right) \tag{12}$$

The output of the decoder is a set of vertices $\mathscr{V}$ and surface information $\mathscr{S}$. From this, a mesh $\mathscr{M} = (\mathscr{V}, \mathscr{F})$ can be obtained using the Delaunay triangulation and determine triangle visibility using $\mathscr{S}$, where the triangle surface score is $s_i^{surf} > 0$ (see Equation 5).

4 EVALUATION AND RESULTS

In the following section we describe our experiment setup and qualitative as well as quantitative results, and analyze the transformer attention maps for model interpretability. An example of input and output is given in Figure 4.

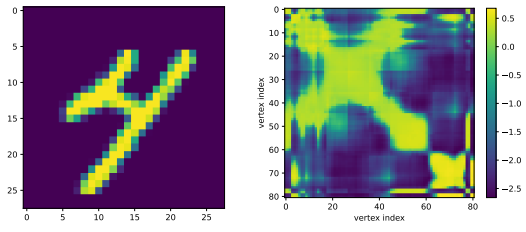
4.1 Experiment Setup

Hidden Dims	256
Encoder Blocks	4
Encoder Attention Heads	4
Encoder MLP Size	256
Decoder Blocks	4
Encoder Decoder Heads	4
Decoder MLP Size	512

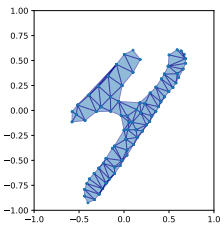
Table 1: Model configuration

We demonstrate our approach by training a Transformer model to reconstruct shapes from the MNIST dataset [Den12] of handwritten digits. Our model has 256 hidden dimensions with an encoder and decoder that both have 4 blocks. See Table 1 for the entire model configuration. In total, our model has $7.52M$ parameters.

We train our model with the AdamW optimizer with $lr = 10^{-4}$, $\beta_1 = 0.999$, and $\beta_2 = 0.98$ for 150 epochs; training time was roughly 6 hours on an Nvidia A40 GPU. We used a cosine learning rate schedule with 1,500 warmup steps. This means the learning rate is slowly increased in the beginning, and gets decreased over time until it reaches zero. We have also experimented with hard resets, where the learning rate gets repeatedly reduced to zero and is then reset to the original learning rate [LH16]. However, hard resets were



(a) Input (b) Surface matrix $\mathscr{S}$

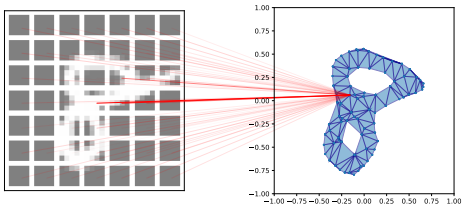


(c) Final triangulation

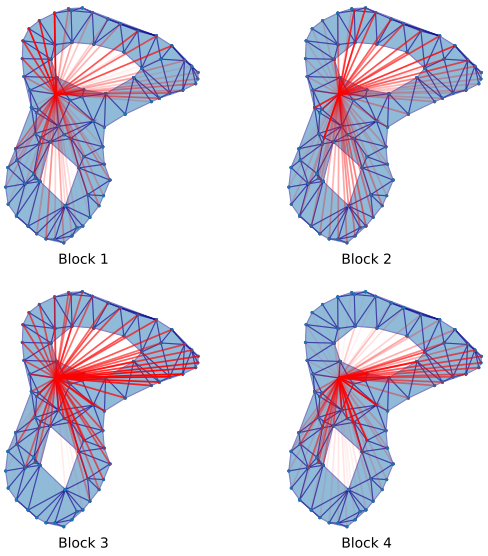
Figure 4: Given an input image (4a), our method predicts a set of vertices $\mathscr{V}$ and surface information $\mathscr{S}$ (4b) that encodes edge visibility. From this, a mesh (4c) can be obtained using the Delaunay triangulation.

not necessary when we used the non-linear MLP in the geometry head of the decoder.

4.2 Model Analysis



(a) Cross Attention



(b) Self Attention

Figure 5: Transformers can learn meaningful representations of meshes; we visualize attention values of one vertex, the alpha value of the red lines corresponds to the attention weight. Cross attention (5a) learns the relationship between vertex positions and image patches while self attention (5b) learns the connectivity of the shape.

We visualize cross- and self-attention weights to analyze if the Transformer learns meaningful latent embeddings. Figure 5 depicts the attention maps learned in the final transformer block before the geometry head.

Cross-attention (Figure 5a) shows how a vertex attends to different parts of the input image. The attention tends to be strongest for the image patches corresponding to the location of the vertex. However, there are also many samples where this is not case. We suspect that the transformer does not need to learn input patch attention for every vertex, due to being able to generate this information from latent space information of other vertices.

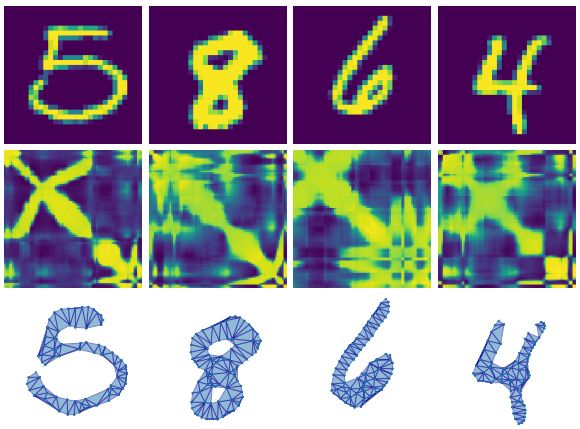


Figure 6: Qualitative examples of input images (top), corresponding surface matrices (middle) and predicted output meshes (bottom) of the test set. While reproducing the shape, the model may sometimes fail to reproduce fine details, e.g. the hole in the 6, or the upper right part of the 4.

The attention map of self-attention (Figure 5b) classifies all possible edges between the vertices as either visible or invisible. As depicted in the figure, the attention between two vertices is strongest when they are neighbors. We theorize that self-attention acts as a regularization that forces neighboring vertices to be similar in latent space. This latent classification map is then used by the geometry head to classify the visibility.

Some more qualitative examples are shown in Figure 6. We also investigate the ability of the model to generalize to shapes of unseen topologies by predicting meshes for samples of the EMNIST dataset of letters, see Figure 7.

4.3 Quantitative Results

We quantitatively evaluate two key aspects of the model: we determine the model's capability of faithfully reproducing the input via pixel-wise binary classification of a hard binary rasterization of the triangulation, and we analyze the generated triangle quality with a triangle shape metric.

Our model achieves 95.596% pixel accuracy for foreground and background pixels on the unseen MNIST test dataset. For analyzing triangle quality, we use the f_{shape} metric [Knu03] from finite element analysis, which yields 1.0 for equilateral and 0.0 for degenerate triangles. This metric is not contained in the training loss formulation. See Figure 8 for the distribution of this metric over all predicted triangles of the dataset; the median of this metric is 0.67, both for the training and the test dataset. On a single NVIDIA A40 GPU, evaluating the model on the MNIST dataset (50,000 samples) takes around 30 seconds and requires around 6.4 GB

GPU memory. Using a batch size of 256, this equals roughly 153*ms* per batch.

5 CONCLUSION

In this paper, we show first results for learning meshes with arbitrary genus with transformers. Previous methods either rely on a template mesh, or generate vertices sequentially which requires to introduce an order of the vertices. In contrast, our decoder is able to jointly represent vertex positions and neighborhood information while being permutation invariant. We developed a self-supervised training pipeline that learns to predict triangle meshes from input images. Our results show that the proposed architecture is able to reproduce these shapes, and is even able to reproduce some shapes with unseen topology.

However, while the method is able to represent meshes with varying topology, it also still has some noteworthy limitations. First of all, given that the vertices are generated all at once rather than sequentially, the number of vertices is always fixed. This is a natural trade off when using a non-sequential architecture. As the surface matrix maps the complete graph of all vertex connections which scales quadratically with the number of vertices. An important line of future research is to efficiently scaling the approach up to larger meshes. This could be accomplished by applying attention hierarchically and locally akin to convolutions in CNNs; an architecture like this would leverage the local nature of topology information to be more scalable. Furthermore, we aim to extend our approach to 3D, which would enable a wide range of applications such as single view reconstruction or generative modeling. A possible extension would involve generating a Delaunay triangulation in 3D space and subsequently classifying tetrahedra as either inside or outside based on the vertex-to-vertex weights encoded within the surface matrix. While the model is very flexible, currently no guarantees about the generated geometry are given; another avenue for future work would be to guarantee certain geometric properties like manifoldness in the approach.

We believe this work is an important first step towards building models that can learn to directly predict meshes with arbitrary topology.

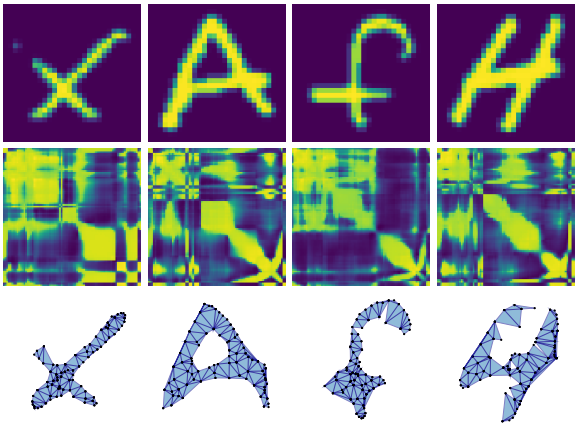


Figure 7: Images from the EMNIST dataset of letters contain shapes of topologies that are not present in the MNIST digits dataset; the model was trained on the digits. It shows the ability to reproduce the shapes of X and A quite faithfully, while in some cases the reconstruction misses parts of the shape, as shown with the letters f and H.

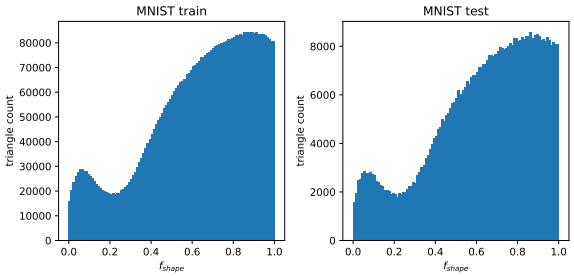


Figure 8: We evaluate the triangle shape metric f_{shape} for all triangles in the training (left) and test (right) dataset. The distribution shows a trend towards more equilaterally shaped triangles.

REFERENCES

[ACB17] Martin Arjovsky, Soumith Chintala, and Léon Bottou. "Wasserstein Generative Adversarial Networks". In: *Proceedings of the 34th International Conference on Machine Learning*. Ed. by Doina Precup and Yee Whye Teh. Vol. 70. Proceedings of Machine Learning Research. PMLR, Aug. 2017, pp. 214–223.

[All+08] Pierre Alliez et al. "Recent Advances in Remeshing of Surfaces". In: *Shape Analysis and Structuring*. Ed. by Leila De Floriani and Michela Spagnuolo. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 53–82. ISBN: 978-3-540-33265-7.

[BC+22] Tarek Ben Charrada et al. "TopoNet: Topology Learning for 3D Reconstruction of Objects of Arbitrary Genus". In: *Computer Graphics Forum* 41.6 (2022), pp. 336–347.

[BDS18] Andrew Brock, Jeff Donahue, and Karen Simonyan. "Large Scale GAN Training for High Fidelity Natural Image Synthesis". In: *CoRR* (2018).

[BH+18] Heli Ben-Hamu et al. “Multi-Chart Generative Surface Modeling”. In: *ACM Transactions on Graphics (TOG)* 37.6 (2018), pp. 1–15.

[Bro+20] Tom Brown et al. “Language Models are Few-Shot Learners”. In: *Advances in Neural Information Processing Systems*. Ed. by H. Larochelle et al. Vol. 33. Curran Associates, Inc., 2020, pp. 1877–1901.

[Cho+16] Christopher B Choy et al. “3D-R2N2: A Unified Approach for Single and Multi-view 3D Object Reconstruction”. In: *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part VIII 14*. Springer. 2016, pp. 628–644.

[Den12] Li Deng. “The mnist database of handwritten digit images for machine learning research”. In: *IEEE Signal Processing Magazine* 29.6 (2012), pp. 141–142.

[Dev+19] Jacob Devlin et al. “BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding”. In: *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. Minneapolis, Minnesota: Association for Computational Linguistics, June 2019, pp. 4171–4186.

[Dos+20] Alexey Dosovitskiy et al. “An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale”. In: (2020).

[Fen+18] Leman Feng et al. “Curved optimal delaunay triangulation”. In: *ACM Trans. Graph.* 37.4 (July 2018). ISSN: 0730-0301.

[FSG17] Haoqiang Fan, Hao Su, and Leonidas Guibas. “A Point Set Generation Network for 3D Object Reconstruction from a Single Image”. In: (2017), pp. 605–613.

[GMJ19] Georgia Gkioxari, Jitendra Malik, and Justin Johnson. “Mesh R-CNN”. In: (2019), pp. 9784–9794.

[Goo+14] Ian Goodfellow et al. “Generative Adversarial Nets”. In: *Advances in Neural Information Processing Systems*. Ed. by Z. Ghahramani et al. Vol. 27. Curran Associates, Inc., 2014.

[Gro+18] Thibault Groueix et al. “AtlasNet: A Papier-Mâché Approach to Learning 3D Surface Generation”. In: *Proceedings IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*. 2018.

[Gul+17] Ishaan Gulrajani et al. “Improved Training of Wasserstein GANs”. In: *Proceedings of the 31st International Conference on Neural Information Processing Systems*. 2017, pp. 5769–5779.

[Hao+24] Zekun Hao et al. *Meshtron: High-Fidelity, Artist-Like 3D Mesh Generation at Scale*. 2024. arXiv: 2412.09548 [cs.GR].

[HJA20] Jonathan Ho, Ajay Jain, and Pieter Abbeel. “Denoising Diffusion Probabilistic Models”. In: *Advances in Neural Information Processing Systems*. Vol. 33. 2020, pp. 6840–6851.

[Kar+18] Tero Karras et al. “Progressive Growing of GANs for Improved Quality, Stability, and Variation”. In: *International Conference on Learning Representations*. 2018.

[Kha+22] Dawar Khan et al. “Surface Remeshing: A Systematic Literature Review of Methods and Research Directions”. In: *IEEE Transactions on Visualization and Computer Graphics* 28.3 (2022), pp. 1680–1713.

[Knu03] Patrick M. Knupp. “Algebraic mesh quality metrics for unstructured initial meshes”. In: *Finite Elements in Analysis and Design* 39.3 (2003), pp. 217–241.

[Lew+20] Mike Lewis et al. “BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension”. In: *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Online: Association for Computational Linguistics, July 2020, pp. 7871–7880.

[LH16] Ilya Loshchilov and Frank Hutter. “SGDR: Stochastic Gradient Descent with Warm Restarts”. In: (2016).

[LHJ19] Guillaume Loubet, Nicolas Holzschuch, and Wenzel Jakob. “Reparameterizing Discontinuous Integrands for Differentiable Rendering”. In: *ACM Transactions on Graphics (TOG)* 38.6 (2019), pp. 1–14.

[Li+18] Tzu-Mao Li et al. “Differentiable Monte Carlo Ray Tracing through Edge Sampling”. In: *ACM Transactions on Graphics (TOG)* 37.6 (2018), pp. 1–11.

[Liu+19] Shichen Liu et al. “Soft Rasterizer: A Differentiable Renderer for Image-Based 3D Reasoning”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)* (Oct. 2019), pp. 7708–7717.

[Mil+20] Ben Mildenhall et al. “NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis”. In: *European Conference on Computer Vision*. 2020, pp. 405–421.

[Nas+20] Charlie Nash et al. “PolyGen: An Autoregressive Generative Model of 3D Meshes”. In: *Proceedings of the 37th International Conference on Machine Learning*. Ed. by Hal Daumé III and Aarti Singh. Vol. 119. Proceedings of Machine Learning Research. PMLR, July 2020, pp. 7220–7229.

[Nic+22] Alex Nichol et al. “GLIDE: Towards Photorealistic Image Generation and Editing withText-Guided Diffusion Models”. In: (2022), pp. 16784–16804.

[Pan+19] Junyi Pan et al. “Deep Mesh Reconstruction from Single RGB Imagesvia Topology Modification Networks”. In: (2019), pp. 9964–9973.

[Par+19] Jeong Joon Park et al. “DeepSDF: Learning Continuous Signed Distance Functionsfor Shape Representation”. In: (2019), pp. 165–174.

[Qi+17] Charles R. Qi et al. “PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017, pp. 652–660.

[Rad+19] Alec Radford et al. “Language Models are Unsupervised Multitask Learners”. In: (2019).

[Rah+19] Nasim Rahaman et al. “On the Spectral Bias of Neural Networks”. In: (2019), pp. 5301–5310.

[Rak+21] Marie-Julie Rakotosaona et al. “Differentiable Surface Triangulation”. In: *ACM Trans. Graph.* (2021).

[Rav+20] Nikhila Ravi et al. “Accelerating 3D Deep Learning with PyTorch3D”. In: (2020).

[Rom+22] Robin Rombach et al. “High-Resolution Image Synthesis With Latent Diffusion Models”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2022, pp. 10684–10695.

[Sah+22] Chitwan Saharia et al. “Photorealistic Text-to-Image Diffusion Models with Deep Language Understanding”. In: *Advances in Neural Information Processing Systems*. Ed. by S. Koyejo et al. Vol. 35. Curran Associates, Inc., 2022, pp. 36479–36494.

[SD+15] Jascha Sohl-Dickstein et al. “Deep Unsupervised Learning using Nonequilibrium Thermodynamics”. In: *Proceedings of the 32nd International Conference on Machine Learning*. Ed. by Francis Bach and David Blei. Vol. 37. Proceedings of Machine Learning Research. Lille, France: PMLR, July 2015, pp. 2256–2265.

[SE19] Yang Song and Stefano Ermon. “Generative Modeling by Estimating Gradients of the Data Distribution”. In: *Proceedings of the 33rd International Conference on Neural Information Processing Systems*. 2019, pp. 11918–11930.

[She12] Jonathan Richard Shewchuk. “Unstructured mesh generation”. In: *Combinatorial Scientific Computing* 12.257 (2012), p. 2.

[Sid+23] Yawar Siddiqui et al. “MeshGPT: Generating Triangle Meshes with Decoder-Only Transformers”. In: *arXiv preprint arXiv:2311.15475* (2023).

[Tan+20] Matthew Tancik et al. “Fourier Features Let Networks Learn High Frequency Functions in Low Dimensional Domains”. In: *Advances in Neural Information Processing Systems*. Ed. by H. Larochelle et al. Vol. 33. Curran Associates, Inc., 2020, pp. 7537–7547.

[Vas+17] Ashish Vaswani et al. “Attention is All you Need”. In: *Advances in Neural Information Processing Systems*. Ed. by I. Guyon et al. Vol. 30. Curran Associates, Inc., 2017.

[VFJ15] Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. “Pointer Networks”. In: *Proceedings of the 28th International Conference on Neural Information Processing Systems-Volume 2*. 2015, pp. 2692–2700.

[Wan+18] Nanyang Wang et al. “Pixel2Mesh: Generating 3D Mesh Models from Single RGB Images”. In: *ECCV* (2018), pp. 52–67.

[Wen+19] Chao Wen et al. “Pixel2Mesh++: Multi-View 3D Mesh Generation via Deformation”. In: *Proceedings of the IEEE/CVF international conference on computer vision*. 2019, pp. 1042–1051.

[WLH16] Thomas Wiemann, Kai Lingemann, and Joachim Hertzberg. “Optimizing Triangle Mesh Reconstructions of Planar Environments”. In: *IFAC-PapersOnLine* 49.15 (2016). 9th IFAC Symposium on Intelligent Autonomous Vehicles IAV 2016, pp. 218–223. ISSN: 2405-8963.

[Wu+15] Zhirong Wu et al. “3D ShapeNets: A Deep Representation for Volumetric Shapes”. In: *CVPR* (2015).

[Wu+16] Jiajun Wu et al. “Learning a Probabilistic Latent Space of Object Shapes via 3D Generative-Adversarial Modeling”. In: *Advances in Neural Information Processing Systems*. Ed. by D. Lee et al. Vol. 29. Curran Associates, Inc., 2016.

[Yan+18] Yaoqing Yang et al. “FoldingNet: Point Cloud Auto-encoder via Deep Grid Deformation”. In: *Computer Vision and Pattern Recognition Conference* (2018), pp. 206–215.

[Zha+21] Cheng Zhang et al. “Path-Space Differentiable Rendering”. In: *ACM Transactions on Graphics (TOG)* 40.4 (2021), pp. 1–15.