TetGAN: A Convolutional Neural Network for Tetrahedral Mesh Generation

William Gao[1]
wmg@uchicago.edu
April Wang[1]
aprilyanli38@gmail.com
Gal Metzer[2]
gal.metzer@gmail.com
Raymond A. Yeh[3]
rayyeh@purdue.edu
Rana Hanocka[1]
ranahanocka@uchicago.edu

[1] University of Chicago
Chicago, IL, USA
[2] Tel Aviv University
Tel Aviv-Yafo, Israel
[3] Purdue University
West Lafayette, IN, USA

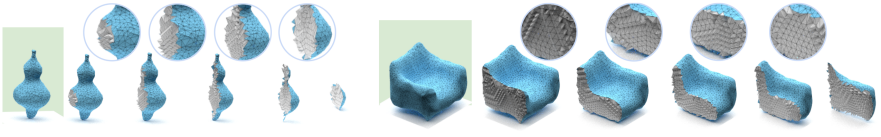


Figure 1: TetGAN **synthesizes novel tetrahedral meshes** with solid interiors.

Abstract

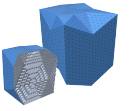
We present TetGAN, a convolutional neural network designed to generate tetrahedral meshes. We represent shapes using an irregular tetrahedral grid which encodes an occupancy and displacement field. Our formulation enables defining tetrahedral convolution, pooling, and upsampling operations to synthesize explicit mesh connectivity with variable topological genus. The proposed neural network layers learn deep features over each tetrahedron and learn to extract patterns within spatial regions across multiple scales. We illustrate the capabilities of our technique to encode tetrahedral meshes into a semantically meaningful latent-space which can be used for shape editing and synthesis. Our project page is at https://threedle.github.io/tetGAN/.

1 Introduction

We approach learning to synthesize and edit 3D shapes through a neural framework designed for *tetrahedral* meshes. Tetrahedral meshes are used widely across a myriad of applications, such as articulated shape deformations [21, 49], solid modeling [6], scientific computing [47], and physics-based simulations [19, 22]. However, deep learning on tetrahedral meshes is only just starting to develop [8, 43].

We propose a framework for generating tetrahedral meshes, called TetGAN. Our system learns to produce *novel* and diverse tetrahedral meshes from random noise. Simultaneously, TetGAN provides *controlled* manipulation of existing tetrahedral meshes through

latent-encoding capabilities. Our framework trains an encoder to map tetrahedral meshes into a semantically meaningful latent space. This enables performing latent space interpolation or arithmetic operations for effective 3D shape manipulations.



TetGAN leverages an irregular grid of tightly packed tetrahedra within a bounding cuboid to create a spatial structure for learning local correlations in 3D. Akin to a pixel in an image, each tetrahedron is the basic building block for learning over the network input. To represent a 3D shape, we encode each tetrahedron with an *occupancy* value and 3-dimensional *displacement* vector to the closest point on the shape surface. This enables extracting a tetrahedral mesh by removing unoccupied tetrahedra, and displacing vertices on triangles on the surface *boundary*. This representation combines the advantages of implicit representations to handle varying topology, and explicit meshes to obtain a high level of detail efficiently.

The irregular tetrahdral grid has recently started to gain traction as an alternative neural representation for 3D geometry [33], pioneered by DefTet and DMTet [8, 43]. Using the irregular tetrahedral grid, topological holes can be inserted anywhere easily by removing unoccupied tetrahedra in the grid. Tetrahedral meshes are able to *deform* each triangle by individually displacing vertices since deformed triangles remain planar *and* connected to adjacent faces. Deforming the boundary vertices to coincide with the learned underlying shape will result in a smooth surface with varied topology. This seemingly simple property is non-trivial to obtain with 3D cubic voxel grids (i.e. a hexahedral mesh). Displacing 4-points in a quadrilateral does not necessarily preserve planarity between adjacent quads.

We learn to represent local spatial relationships in 3D shapes (such as corners, curves, and planes), through carefully designed neural network layers. We use a *tetrahedral convolution* layer which shares weights across the entire spatial field, in order to encode local shape cues. Beyond local regions, we progressively incorporate additional 3D *context* through *tetrahedral pooling*, reducing the spatial resolution of the input. The original spatial resolution is restored through *upsampling* layers. The TetGAN layers contain an inductive bias which leverages and learns from features across multiple spatial resolutions.

Our TetGAN framework supports and benefits from both *latent encoding* capabilities during inference, and unsupervised *adversarial* losses during training. We train our model using distinct VAE and GAN objectives, which is different than typical use of VAE with GAN [7, 53]. In this work, the VAE training objective aims for tetrahedral meshes to be encoded and then exactly decoded using a reconstruction loss. Then, during inference the encoder maps a given input shape into the learned latent space, enabling performing latent-based editing operations. The GAN training objective aims to generate new tetrahedral meshes from noise that are not part of the training dataset. We achieve this by sampling a random latent code from a unit Gaussian distribution, used as input to the same decoder to produce a tetrahedral mesh using an adversarial loss (without any reconstruction loss).

2 Related Work

Initially, convolutional neural networks built for mesh surfaces were focused on shape analysis tasks such as classification and segmentation [13, 17, 27, 32]. Later, techniques proposed synthesizing vertex locations over a surface mesh with fixed topology [9, 14, 15, 29, 48]. Though meshes can efficiently represent complex geometries, generating meshes with varied genus is challenging [38]. Only a handful of works directly generate or modify surface mesh connectivity [34, 40, 42]. Other techniques combine fixed genus mesh parts to form a more

complex topology, such as a composition of primitives [9, 37, 46], parametric patches [11], or assembly of parts [52]. Cubic voxel-based techniques can generate shapes with varied genus [5], however planar-face hexahedra cannot be displaced freely.

Implicit Representations. Shapes can also be synthesized using implicit representations, which predict a signed distance or an occupancy value per input point [3, 31, 36, 51]. Implicit representations can generate shapes with any topological genus since the explicit mesh surface is extracted in post-process. ShapeGAN [26] employs DeepSDF [36] as a generator to model a signed distance function, and a discriminator which evaluates the signed distance field produced by a batch of samples alongside latent information. The generated SDF values from the generator are arranged into a voxel volume or input to PointNet.

Tetrahedral meshes are widely used in Finite Element Method (FEM) simulations [19, 35]. Computing realistic and high-quality deformations of articulated shapes commonly relies on a tetrahedral mesh [21, 49]. We use the tool QuarTet [1] to generate the irregular grid of tetrahedra which we use to define the TetGAN convolution, pooling, and upsampling layers. Our use of a tetrahedral grid draws inspiration from DefTet [8], which learned to reconstruct tetrahedral meshes from point clouds or 2D images. Recently, Deformable Marching Tets (DMTet) [43], learns to reconstruct high-resolution surface meshes from a coarse voxel input.

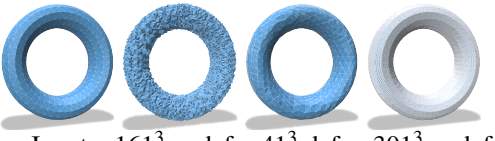


Figure 2: Input torus represented with different volumetric representations. Left to right: i) occupancies extracted from a 161^3 tetrahedralized cube (undeformed), ii) a 41^3 deformable tetrahedral grid, and iii) voxelized torus at 301^3 extracted with marching cubes.

One of the key differences between our tetrahedral grid representation and that of DefTet [8] and DMTet [33, 43] is how we predict and utilize displacements. In this work, our tetrahedral grid contains a vector-valued displacement to the underlying surface at each tetrahedron's centroid, which we refer to as a *deformation field*. Deformation fields are a rich representation of the explicit underlying geometry, which provides shape cues over the entire grid. The objective of encoding and predicting a deformation field even aids the network in learning to predict more accurate occupancy fields, suggesting that deformation fields are an effective neural shape representation. After training is complete, we leverage the network-predicted deformation field to *filter* incorrect occupancy predictions and infer weights for *smoothing* the predicted surface geometry.

While prior works apply the tetrahedral grid representation to the tasks of tetrahedral *reconstruction* (DefTet) and mesh super-resolution (DMTet), our method provides *generative* capabilities, a fundamentally different objective. TetGAN enables sampling novel shapes from noise, latent space interpolations, and shape editing, none of which are provided by DefTet/DMTet (nor by inverse rendering [33]). TetGAN achieves this using a novel CNN for tetrahedral meshes, with convolution/pooling blocks that are distinct from DefTet/DMTet components and tailored for the task of generation (inspired by 2D CNNs [23, 24, 30]). Our pooling and upsampling leverages a tetrahedral subdivision structure to learn features over various receptive fields. In a similar spirit, SubDivNet [17] learns using a loop subdivision structure for performing analysis tasks (e.g. classification) over *surface* meshes.

Contrast to Voxels. Works that synthesize shapes using implicits may opt to predict over a voxel grid [26, 31]. However, the benefits afforded by our deformation field are not trivially amenable to 3D cubic (hexahedral) voxels. Freely displacing the four vertices per-face

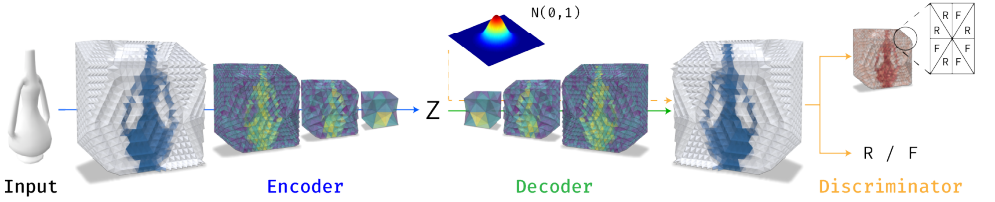
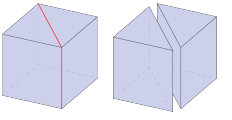


Figure 3: TetGAN overview. We represent the input mesh using a deformation field over a tetrahedral grid. The **encoder** produces a latent vector $\boldsymbol{z}$, which is used as input to a **decoder** which performs a series of convolutions and upsampling operations. The two **discriminators** learn to classify whether the input shape is fake (generated) or real at the patch/global level.



will break the planar-face property. Alternatively, fitting higher-order curved patches to 4-point sets does not guarantee connectivity between adjacent cells. Moreover, cubic cells do not trivially divide into tetrahedra, and volumetric meshing is an open active area of research [18, 19, 44]. Due to the inherent challenges in converting to tetrahedral meshes (e.g. inevitable trade-offs between geometric accuracy and robustness), several works propose bypassing tetrahedral meshing altogether for a particular application [41, 45]. Compared to cubic voxels, tetrahedral meshes are able to more efficiently adapt to complex geometries, such as high curvature regions as shown in Fig. 2. Representing a torus using different volumetric representations (with error under a threshold), requires 301^3 undeformed cubic voxels, compared to 161^3 undeformed tetrahedra or 41^3 deformed tetrahedra.

3 Method

We design a generative model over a packed tetrahedral grid to define a coarse inside / outside representation of a shape (i.e., an occupancy). This grid directly represents a tetrahedral mesh with no additional processing and allows an isosurface to be trivially extracted. We also define a displacement per-vertex in the grid to the underlying shape surface, which enables smoothing the coarse representation to obtain a higher fidelity result.

To learn over this representation, we propose TetGAN, a deep architecture which leverages and fuses ideas from both generative adversarial neural networks (GAN) [10] and Variational Autoencoders (VAE) [25]. First, an encoder performs a series of convolution and pooling layers (on a tetrahedral grid) to aggregate the spatial information over the 3D shape into a latent vector. Next, a decoder processes the latent vector through a series of convolution and upsampling layers to reconstruct the 3D shape. This stage is supervised by ground truth features computed over the training data. Finally, to improve the diversity and smoothness of the learned latent space, we make use of an additional adversarial training, where the VAE decoder acts as the generator of a GAN. This enables TetGAN to directly learn more point samples inside the latent space. To synthesize a new tetrahedral mesh, we sample from the learned latent space and use the decoder to generate the occupancy and displacement fields. The overall model is illustrated in Fig. 3.

3.1 Tetrahedral grid framework

Tetrahedral grid. Our framework represents shapes using an occupancy and a displacement encoded within an irregular grid of tetrahedra. First, we compute an initial tetrahedralization

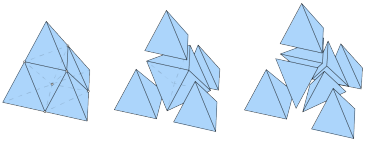


Figure 4: Subdivision structure.

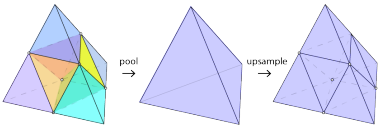


Figure 5: Pooling and upsampling operators.

of a cube using QuarTet [1] to obtain a tetrahedral grid $\mathcal{T} = \{\boldsymbol{T}_1, \dots \boldsymbol{T}_K\}$ consisting of K tetrahedra. Each tetrahedron $\boldsymbol{T}_k$ is represented by its four vertices: $\{v_{k_a}, v_{k_b}, v_{k_c}, v_{k_d}\}$, where each $v_{k_i} \in \mathbb{R}^3$. To represent an input surface mesh $\boldsymbol{M}$, we first normalize it to fit inside the unit cube. We calculate the occupancy value (inside or outside) at each tetrahedron's centroid. We also calculate the displacement from the tetrahedron centroid to the closest point on the mesh surface, i.e. the displacement field.

Tetrahedral convolution. Despite being an irregular grid $\mathcal{T}$, a unique structure naturally admits learning a convolution. Every tetrahedron in the grid is adjacent to exactly four tetrahedra, which defines the spatial support of the convolutional filter. Specifically, for each tetrahedron $\boldsymbol{T}_k \in \mathcal{T}$, we perform a weighted sum of $\boldsymbol{T}_k$ and its four adjacent neighbors $\boldsymbol{T}_j \in \mathcal{T}$ (i.e. $\boldsymbol{T}_k$ and $\boldsymbol{T}_j$ share a triangle face). A convolution on the tetrahedral grid is defined as $\boldsymbol{W}_0 \cdot \phi(\boldsymbol{T}_k) + \sum_{j \in \mathcal{N}_k} \boldsymbol{W}_j \cdot \phi(\boldsymbol{T}_j)$, where $\mathcal{N}_k$ denotes the set of adjacent neighbors of $\boldsymbol{T}_k$, $\boldsymbol{W}$ denotes the convolution weights, and $\phi(\boldsymbol{T}_k)$ denotes the features for tetrahedron $\boldsymbol{T}_k$. The ordering of the four neighbors in the convolution is prescribed by the tetrahedral grid.

Subdivision structure for pooling and upsampling. To perform pooling and upsampling on the tetrahedral grid, we need to first define how to down/upsample, for which we introduce a subdivision structure. For any $\boldsymbol{T}_j \in \mathcal{T}$, we can subdivide it into eight tetrahedra, represented as a *supercell* $\mathcal{S}_j = \{\boldsymbol{T}_k\}$. A supercell is created by adding vertices at the midpoints of each edge. Starting with a low resolution grid $\mathcal{T}^{(1)}$, we produce successive grids $\mathcal{T}^{(2)}, \mathcal{T}^{(3)}, \dots, \mathcal{T}^{(N)}$ by subdividing each tetrahedron. Naturally, pooling of features ϕ from $\mathcal{T}^{(n)}$ to $\mathcal{T}^{(n-1)}$ is defined as

$$\phi^{(n-1)}(\boldsymbol{T}_j^{(n-1)}) = \mathbf{F}\left(\left\{\phi^{(n)}(\boldsymbol{T}_k^{(n)}) \quad \forall\, \boldsymbol{T}_k^{(n)} \in \mathcal{S}_j^{(n-1)}\right\}\right), \tag{1}$$

where $\mathbf{F}$ is a symmetric aggregation function (either a max or average).

To upsample features $\phi^{(n)}$ from $\mathcal{T}(n-1)$ to $\mathcal{T}^{(n)}$, one may define techniques analogous to those in 2D computer vision. We propose an alternative method analogous to interpolation upsampling. The upsampled features of $\boldsymbol{T}_k^{(n)} \in \mathcal{S}_j$ are an average of the features of the neighbors of its parent $\boldsymbol{T}_j^{(n-1)}$, weighted by the inverse distance between centroids:

$$\phi^{(n)}(\boldsymbol{T}_k^{(n)}) = \sum_{\ell \in \mathcal{N}(\boldsymbol{T}_j^{(n-1)})} \frac{\alpha_1}{\left|\left| c_\ell^{(n-1)} - c_k^{(n)} \right|\right|} \phi(T_\ell^{(n-1)}), \tag{2}$$

where α_1 is a scalar that normalizes the coefficients to sum to one. For an illustration of the downsamping and upsampling processes, see Fig. 5

3.2 TetGAN

Problem formulation. We represent the 3D shape by inserting it inside the irregular grid $\mathcal{T}^{(N)}$ and computing an occupancy and a displacement per tetrahedron. The occupancy $\boldsymbol{O}$ is given by: $\boldsymbol{O}(\boldsymbol{T}_k^{(N)}) = \mathbf{1}[w_{\boldsymbol{M}}\left(c_k^{(N)}\right) > 0]$, where $w(\cdot)$ gives the winding number of any point

with respect to the surface $\boldsymbol{M}$ and $\mathbf{1}$ denotes the indicator function. Here, $c_k^{(N)}$ denotes the centroid of $\boldsymbol{T}_k^{(N)}$ given by the average coordinate of its four vertices. We define a deformation on a *tetrahedron* $\boldsymbol{T}_k^{(N)}$ as the difference between $c_k^{(N)}$ and its closest point p on $\boldsymbol{M}$. In order to obtain a deformation over the *vertices*, we compute the average displacement over each tetrahedra incident to each vertex $v^{(N)}$, weighted by the inverse distance:

$$\boldsymbol{D}(v^{(N)}) = \sum_{k \in \mathcal{I}(v^{(N)})} \frac{\alpha_2}{\left|\left|v^{(N)} - c_k^{(N)}\right|\right|} \arg\min_{p \in \boldsymbol{M}} \left|\left|c_k^{(N)} - p\right|\right| - c_k^{(N)}, \tag{3}$$

where $\mathcal{I}(v)$ denotes a set of incident tetrahedra indices of vertex v and α_2 again is a normalizing scalar. We directly penalize the deformation per-vertex against the ground-truth deformation field. To represent a *tetrahedral* mesh $\boldsymbol{M}$ we have $\boldsymbol{X} = [\boldsymbol{O}, \boldsymbol{D}]$. $\boldsymbol{O}$ trivially defines a *triangular mesh* consisting of all faces shared by tetrahedra with different occupancies. Our goal is to learn a generative model $p(\boldsymbol{X})$ from which $\boldsymbol{X}$ can be sampled.

Generative model. Our generative model is a hybrid between a VAE and a GAN. Starting with a VAE, we have an encoder $f_{\texttt{enc}}$, a decoder $f_{\texttt{dec}}$, and a prior function $p(\boldsymbol{Z})$. Both encoder and decoder are deep neural networks consisting of the proposed tetrahedral convolution layers; exact architecture details are deferred to the supplemental. To generate a new mesh during inference, we can easily sample a latent vector $\boldsymbol{Z} \sim p(\boldsymbol{Z})$ and pass it through the decoder $f_{\texttt{dec}}(\boldsymbol{Z})$. We train the VAE using binary cross-entropy on the occupancies and mean squared loss on the deformations. To enable a more densely sampled and smoothly varying latent space, we additionally include global and local (WGAN-GP [12]) adversarial losses. This introduces two discriminator networks $f^l_{\texttt{dis}}$ and $f^g_{\texttt{dis}}$, which operate over local and global attributes. The local discriminator is a fully convolutional *patch discriminator* [28] which outputs a 'real' or 'fake' score per tetrahedron. On the other hand, the global discriminator makes use of pooling layers to aggregate information across the entire grid into a 'real' or 'fake' score.

Deformation-Field Weighted Laplacian Smoothing. We apply Laplacian smoothing to the output mesh using weights computed with the predicted deformation field. Specifically, we first update the vertices by applying the deformation field over them, i.e. $\hat{v}_i^{(N)} = v_i^{(N)} + D(v_i^{(N)})$. Next, these displaced vertices are smoothed by performing a weighted average with their neighbors:

$$\tilde{v}_i^{(N)} = \beta \cdot \hat{v}_i^{(N)} + (1-\beta) \cdot \sum_{\hat{v}_j^{(N)} \in \mathcal{N}(\hat{v}_i^{(N)})} \left|\cos\left(\boldsymbol{D}(v_i^{(N)}), \boldsymbol{D}(v_j^{(N)})\right)\right| \cdot \hat{v}_j^{(N)} \tag{4}$$

where cos denotes the cosine similarity, $\mathcal{N}(v_i^{(N)})$ denotes the neighboring vertices of $\hat{v}_i^{(N)}$ and $\beta \in [0,1]$ is a hyperparameter that controls how much influence the original position of $\hat{v}_i^{(N)}$ exerts on its smoothed position $\tilde{v}_i^{(N)}$.

Deformation field filtering. We observe that for tetrahedra that are truly on the surface of the underlying shape, the corresponding deformation should be relatively small in magnitude. Thus, we can use the predicted deformation field to filter incorrect occupancies. In particular, we compute the average norm of surface tetrahedra deformations in the dataset:

$$\mu = \frac{1}{|D|} \sum_{\boldsymbol{M} \in D} \frac{1}{|\boldsymbol{M}|} \sum_{\substack{T_k \in \boldsymbol{T}, \boldsymbol{O_M}(T_k)=1 \\ \exists T_j \in \mathcal{N}_k : \boldsymbol{O_M}(T_j)=0}} ||\boldsymbol{D}(T_k)|| \tag{5}$$

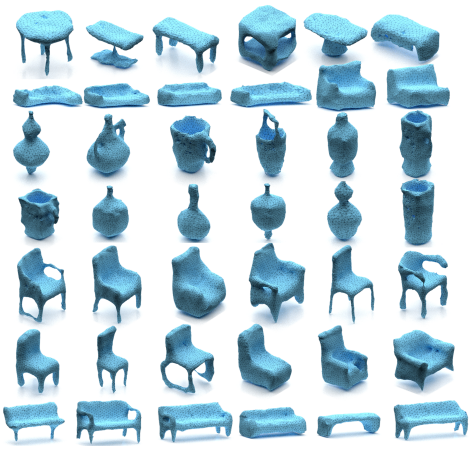


Figure 6: Novel random samples from TetGAN learned latent distribution.

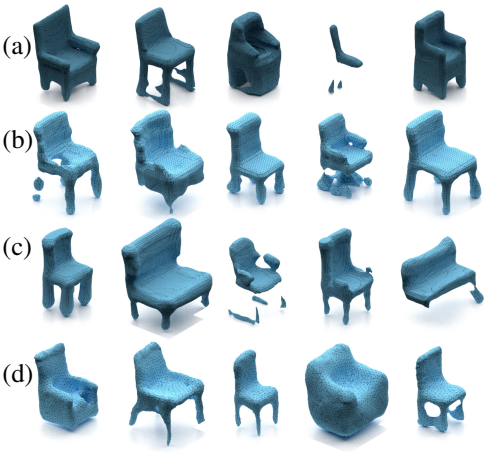


Figure 7: Novel random samples from (a) OccNet, (b) ShapeGAN (VAE), and (c) ShapeGAN, and (d) TetGAN.

where $|\boldsymbol{M}|$ gives the number of occupied tetrahedra on the surface of $\boldsymbol{M}$. During inference (but not during training), we simply discard surface tetrahedra with a deformation larger than $\gamma\mu$ where γ is a tunable parameter.

4 Experiments

In our experiments, we assess the ability of the encoder to effectively map input shapes to the learned latent space, and the decoder to accurately reconstruct them. Next, we evaluate the variety of the generated shape samples when sampling from the learned latent distribution. We compare TetGAN against implicit networks [26, 31] quantitatively using Frechet Inception Distance (FID). Third, we qualitatively explore the latent space through latent space arithmetic and interpolation. Finally, we conduct ablation studies on our own system to quantify the impact of the proposed components, e.g. without adversarial losses. Note that there are no existing works that take tetrahedral meshes as input directly. Therefore, when comparing to baselines, we use their respective choice of discretization.

Dataset. We evaluate TetGAN on five ShapeNet [2] categories (chair, sofa, benches, table, and car), and vases from COSEG [50].

To prepare the training and validation data, we use a randomized $5/1$ train/validation split on the dataset. We process the ShapeNet models to be manifold and watertight using the method by Huang et al. [20], and normalized to a unit bounding cube. We compute occupancy and deformation features for each shape as specified in Sec. 3.2.

4.1 Qualitative results

Visual Sample Quality. To demonstrate the generative capabilities of TetGAN, in Fig. 6 we show random samples over the categories of chair, sofa, table, benches, and vase. These samples are generated unconditionally (i.e. purely from noise). TetGAN is trained separately for each category. We observe that TetGAN is capable of generating a diverse set of

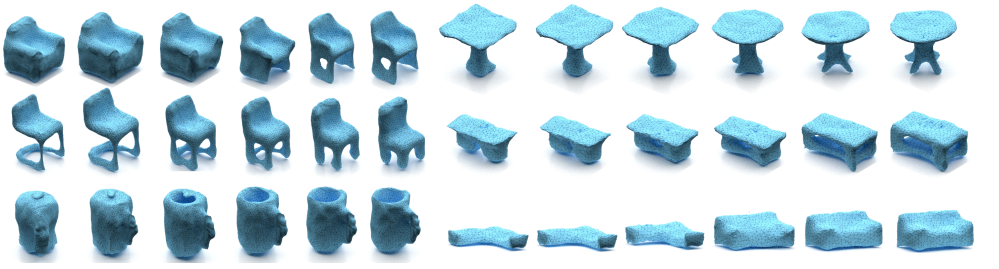


Figure 8: TetGAN's latent space interpolation for different categories.

shapes with varied topology. We also show qualitative comparison with the OccNet [31] and ShapeGAN [26] in Fig. 7.

Latent Space Exploration. We experiment with the latent space TetGAN learns over shapes. In Fig. 8 we demonstrate TetGAN's ability to produce semantically meaningful intermediates between two shapes. We further demonstrate TetGAN encodes high-level semantics of objects and parts through latent space arithmetic.

As illustrated in Fig. 9, the difference between two meshes can be added to another mesh resulting in a completely new object. For example, in the first row, by subtracting the encodings of two similar chairs (in purple and yellow backgrounds) with one main difference (in the chair legs), we are able to transfer that difference onto a third chair (in pink) resulting in a brand-new chair (in green).

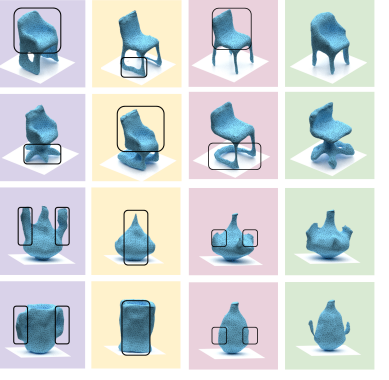


Figure 9: Latent space arithmetic between **mesh A** - mesh B + mesh C = mesh D. The highlighted areas indicate: a part that is (**added**), removed, and where it is inserted.

Utility of TetGAN Meshes. We inspect the synthesized/interpolated meshes by slicing (Fig. 1 and the supplemental), and find that our network predicts meshes with a solid interior. Since these meshes also inherit a high tetrahedral quality from the underlying grid, they are useful for performing engineering simulations (see supplemental).

4.2 Quantitative results

Tab. 1 contains quantitative metrics evaluating reconstruction and generation of chairs. Note that all metrics evaluate *surface quality* since baseline generative methods, unlike TetGAN, do not produce volumetric meshes. See supplemental for additional quantitative evaluation.

Reconstruction. We evaluate how accurately TetGAN reconstructs shapes on a test set using Chamfer Distance as a metric for accuracy. Tab. 1 shows the Chamfer distance between the reconstructed and the ground truth objects as well as the number of faces in the reconstructed object of TetGAN and baseline works. We observe that both OccNet and the VAE version of ShapeGAN beat TetGAN in reconstruction. However, for a fair comparison, we simplify OccNet generated meshes to a comparable number of faces as those produced by TetGAN. In this setting, TetGAN outperforms OccNet in reconstruction. We also note

that, although TetGAN does not beat ShapeGAN (VAE) this metric, reconstruction quality does not necessarily correlate with visual sample quality.

Variety. We evaluate the diversity of TetGAN's shape generation by measuring the most similar pairs in a set of generated shapes. We generate k pairs of shapes and take the average Chamfer distance from the n most similar pairs. In our experiments, we use $k = 250$ and $n = 25$. We run the experiment 5 times to account for variance. We observe that TetGAN produces more varied shapes than both versions of ShapeGAN and comes close to the variety of OccNet, beating the simplified version.

Model	Recon. (Chamfer)↓	Avg. # Faces↓	FID↓	Variety↑
OccNet	0.0015	86901	**1.06**	**0.0041**
ShapeGAN (GAN)	-	25837	3.38	0.0036
ShapeGAN (VAE)	**0.0011**	6115	8.66	0.0017
OccNet*	0.019	5500	1.59	0.0039
ShapeGAN (GAN*)	-	5500	3.87	0.0038
ShapeGAN (VAE*)	**0.0012**	5500	9.25	0.0016
TetGAN – HR/wLS	0.004	5263	2.18	**0.0040**
TetGAN – HR/No wLS	0.0035	5263	**1.35**	0.0028
TetGAN – LR/wLS	0.0032	**3062**	5.01	0.0038
TetGAN – LR/No wLS	0.0029	**3062**	2.09	0.0033

Table 1: HR/LR refer to grid resolution of $(61^3)/(41^3)$. wLS refers to our deformation-field weighted Laplacian smoothing. * indicates mesh simplification of outputs. Note that these metrics are computed over surfaces since baseline methods do not generate volumes. Even so, TetGAN outperforms alternatives at generation in comparable object resolutions (i.e. number of faces).

Frechet Inception Distance. We use **FID** [16] between a set of generated shapes and the hold-out set to measure quality of generated shapes. To compute this distance, we use deep features from a PointNet++ [39] architecture trained for classification. We observe that at a high grid resolution, TetGAN without Laplacian smoothing comes close to the FID that OccNet achieves and beats both versions of ShapeGAN. TetGAN again beats the simplfied version of OccNet. TetGAN at low grid resolution also produces shapes of reasonable quality and FID score. We observe that Laplacian smoothing, although produces desirable results, harms the FID score. This is likely due to the smoothing producing unnaturally thin pieces.

4.3 Ablation Study

Adversarial ablations. We evaluate the effectiveness of the components in TetGAN. TetGAN performs best using the full system, although the global discriminator appears far more important to the quality of generated results. Visually, as seen in Fig. 10, removing the local discriminator (row 4) causes floating artifacts. Removing the global discriminator (row 5) ruins the generative ability of TetGAN. We find that the global discriminator is crucial to learning the global structure of shapes. This is supported by a sharp increase in FID (23.35 without the global discriminator, only 12.27 without the local discriminator).

Graph convolutions. In Fig. 10, row (b), we train using the GCN used in DefTet [8]. Their method trains a GCN to predict displacements over the vertices of a surface extracted from the ground-truth meshes. In this setting, we observe that the predicted deformations tend to be quite small and not well-suited for a noisy occupancy field. In Fig. 10 row (c), we learn a tetrahedral GCN (learning the same weight for each the four neighbors), and observe significant quality degradation.

Deformation and smoothing. We experimented with different methods of smoothing the output of TetGAN. As seen in Fig. 11, the learned displacements were often were not sufficient to create a smooth mesh. However, combining the vertex displacements with an additional weighted Laplacian smoothing (wLS) procedure yields desirable results, smoother

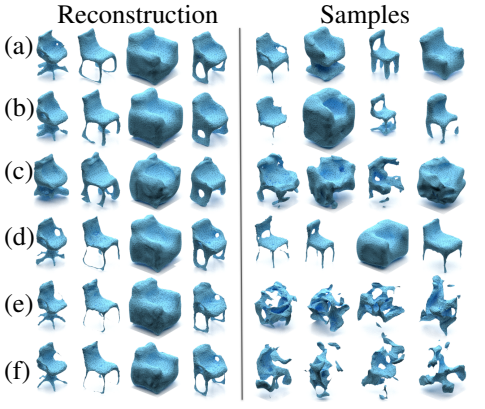


Figure 10: Ablations. Top to bottom: (a) full TetGAN, (b) surface GCN, (c) tetrahedral GCN, (d) no patch discriminator, (e) no global discriminator, (f) VAE only.

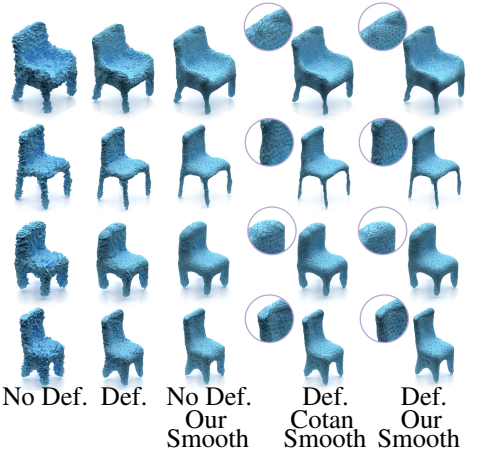


Figure 11: Our deformation-field weighted smooth (Def. Our Smooth) improves the quality of the final result.

and better preservation of detail compared to LS alone. Comparing the last two columns, we also observe that our wLS creates sharper features compared to cotangent LS.

5 Conclusion

We presented a technique for generating tetrahedral meshes, which learns to encode occupancy and displacement field over an irregular grid of tetrahedra. This formulation enables building neural networks capable of tetrahedral generation and manipulation. Our approach combines advantages from both adversarial learning and variational encoding, enabling learning semantically meaningful latent spaces which can synthesize novel and diverse shapes with varied topology using a small number of faces. TetGAN utilizes a unique spatial structure for 3D shapes to build layers which contain a powerful *inductive bias*.

A current limitation of our system is that volumetric grids have a large memory footprint, especially compared to images. An interesting direction for future work involves incorporating a hierarchical subdivision scheme, which would enable selectively adding resolution where needed. In the future we are interested in exploring a technique for generative tetrahedral meshes using an alternative tetrahedral meshing strategy, such as TetWild [18] or TetGen [44] to learn to predict tetrahedra whose size is adaptive to the volume.

Acknowledgements This work was partially supported by gifts from Adobe Research. This work was supported through the University of Chicago AI cluster resources, services, and staff expertise. We thank Greg Shakhnarovich, Vova Kim, Daniele Panozzo, Haochen Wang, Meitar Shechter, Kfir Aberman, and the *3DL group* for their helpful comments, suggestions, and insightful discussions. We thank Guanyue Rao and Xiaosheng Gao for their help in performing physical simulations. We thank Zihan (Jack) Zhang for assistance running experiments, and the anonymous reviewers for their comments.

References

[1] Robert Bridson and Crawford Doran. Quartet. https://github.com/crawforddoran/quartet, 2013.

[2] Angel X. Chang, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, Jianxiong Xiao, Li Yi, and Fisher Yu. ShapeNet: An Information-Rich 3D Model Repository. Technical Report arXiv:1512.03012 [cs.GR], Stanford University — Princeton University — Toyota Technological Institute at Chicago, 2015.

[3] Zhiqin Chen and Hao Zhang. Learning implicit fields for generative shape modeling. In *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 2019.

[4] Zhiqin Chen, Andrea Tagliasacchi, and Hao Zhang. Bsp-net: Generating compact meshes via binary space partitioning. In *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 2020.

[5] Zhiqin Chen, Vladimir G Kim, Matthew Fisher, Noam Aigerman, Hao Zhang, and Siddhartha Chaudhuri. DECOR-GAN: 3D shape detailization by conditional refinement. In *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 2021.

[6] Barbara Cutler, Julie Dorsey, Leonard McMillan, Matthias Müller, and Robert Jagnow. A procedural approach to authoring solid models. *ACM Transactions on Graphics*, 2002.

[7] Patrick Esser, Robin Rombach, and Bjorn Ommer. Taming transformers for high-resolution image synthesis. In *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 2021.

[8] Jun Gao, Wenzheng Chen, Tommy Xiang, Alec Jacobson, Morgan McGuire, Sanja Fidler, Huan Ling, David Acuna, Karsten Kreis, Seung Kim, et al. Learning deformable tetrahedral meshes for 3D reconstruction. In *Advances in Neural Info. Proc. Systems*, 2020.

[9] Lin Gao, Jie Yang, Tong Wu, Yu-Jie Yuan, Hongbo Fu, Yu-Kun Lai, and Hao Zhang. Sdm-net: Deep generative network for structured deformable mesh. *ACM Transactions on Graphics*, 2019.

[10] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in Neural Info. Proc. Systems*, 2014.

[11] Thibault Groueix, Matthew Fisher, Vladimir G Kim, Bryan C Russell, and Mathieu Aubry. A papier-mâché approach to learning 3D surface generation. In *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 2018.

[12] Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron Courville. Improved training of Wasserstein GANs. In *Advances in Neural Info. Proc. Systems*, 2017.

[13] Rana Hanocka, Amir Hertz, Noa Fish, Raja Giryes, Shachar Fleishman, and Daniel Cohen-Or. MeshCNN: a network with an edge. *ACM Transactions on Graphics*, 2019.

[14] Rana Hanocka, Gal Metzer, Raja Giryes, and Daniel Cohen-Or. Point2Mesh: A self-prior for deformable meshes. *ACM Transactions on Graphics*, 2020.

[15] Amir Hertz, Rana Hanocka, Raja Giryes, and Daniel Cohen-Or. Deep geometric texture synthesis. *ACM Transactions on Graphics*, 2020.

[16] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. GANs trained by a two time-scale update rule converge to a local nash equilibrium. In *Advances in Neural Info. Proc. Systems*, 2017.

[17] Shi-Min Hu, Zheng-Ning Liu, Meng-Hao Guo, Jun-Xiong Cai, Jiahui Huang, Tai-Jiang Mu, and Ralph R. Martin. Subdivision-based mesh convolution networks. *ACM Transactions on Graphics*, 2021.

[18] Yixin Hu, Qingnan Zhou, Xifeng Gao, Alec Jacobson, Denis Zorin, and Daniele Panozzo. Tetrahedral meshing in the wild. *ACM Transactions on Graphics*, 2018.

[19] Yixin Hu, Teseo Schneider, Bolun Wang, Denis Zorin, and Daniele Panozzo. Fast tetrahedral meshing in the wild. *ACM Transactions on Graphics*, 2020.

[20] Jingwei Huang, Hao Su, and Leonidas Guibas. Robust watertight manifold surface generation method for ShapeNet models. *arXiv preprint arXiv:1802.01698*, 2018.

[21] Alec Jacobson, Ilya Baran, Jovan Popovic, and Olga Sorkine. Bounded biharmonic weights for real-time deformation. *ACM Transactions on Graphics*, 2011.

[22] Krishna Murthy Jatavallabhula, Miles Macklin, Florian Golemo, Vikram Voleti, Linda Petrini, Martin Weiss, Breandan Considine, Jerome Parent-Levesque, Kevin Xie, Kenny Erleben, et al. gradSim: Differentiable simulation for system identification and visuomotor control. In *Intl. Conf. on Learning Representations*, 2021.

[23] Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. In *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 2019.

[24] Tero Karras, Miika Aittala, Samuli Laine, Erik Härkönen, Janne Hellsten, Jaakko Lehtinen, and Timo Aila. Alias-free generative adversarial networks. In *Advances in Neural Info. Proc. Systems*, 2021.

[25] Diederik P. Kingma and Max Welling. Auto-Encoding Variational Bayes. In *International Conference on Learning Representations*, 2014.

[26] Marian Kleineberg, Matthias Fey, and Frank Weichert. Adversarial generation of continuous implicit shape representations. In *Conf. of the European Association for Computer Graphics*, 2020.

[27] Alon Lahav and Ayellet Tal. Meshwalker: Deep mesh understanding by random walks. *ACM Transactions on Graphics*, 2020.

[28] Chuan Li and Michael Wand. Precomputed real-time texture synthesis with Markovian generative adversarial networks. In *Proc. European Conf. on Computer Vision*, 2016.

[29] Hsueh-Ti Derek Liu, Vladimir G Kim, Siddhartha Chaudhuri, Noam Aigerman, and Alec Jacobson. Neural subdivision. *ACM Transactions on Graphics*, 2020.

[30] Mehran Mehralian and Babak Karasfi. Rdcgan: Unsupervised representation learning with regularized deep convolutional generative adversarial networks. In *AIAR*, 2018.

[31] Lars Mescheder, Michael Oechsle, Michael Niemeyer, Sebastian Nowozin, and Andreas Geiger. Occupancy networks: Learning 3D reconstruction in function space. In *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 2019.

[32] Francesco Milano, Antonio Loquercio, Antoni Rosinol, Davide Scaramuzza, and Luca Carlone. Primal-dual mesh convolutional neural networks. In *Advances in Neural Info. Proc. Systems*, 2020.

[33] Jacob Munkberg, Jon Hasselgren, Tianchang Shen, Jun Gao, Wenzheng Chen, Alex Evans, Thomas Mueller, and Sanja Fidler. Extracting Triangular 3D Models, Materials, and Lighting From Images. In *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 2022.

[34] Charlie Nash, Yaroslav Ganin, SM Ali Eslami, and Peter Battaglia. Polygen: An autoregressive generative model of 3D meshes. In *Intl. Conf. on Machine Learning*, 2020.

[35] Gilles-Philippe Paillé, Nicolas Ray, Pierre Poulin, Alla Sheffer, and Bruno Lévy. Dihedral angle-based maps of tetrahedral meshes. *ACM Transactions on Graphics*, 2015.

[36] Jeong Joon Park, Peter Florence, Julian Straub, Richard Newcombe, and Steven Lovegrove. DeepSDF: Learning continuous signed distance functions for shape representation. In *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 2019.

[37] Despoina Paschalidou, Angelos Katharopoulos, Andreas Geiger, and Sanja Fidler. Neural parts: Learning expressive 3D shape abstractions with invertible neural networks. In *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 2021.

[38] Songyou Peng, Chiyu "Max" Jiang, Yiyi Liao, Michael Niemeyer, Marc Pollefeys, and Andreas Geiger. Shape as points: A differentiable Poisson solver. In *Advances in Neural Info. Proc. Systems*, 2021.

[39] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3D classification and segmentation. In *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 2017.

[40] Marie-Julie Rakotosaona, Noam Aigerman, Niloy J Mitra, Maks Ovsjanikov, and Paul Guerrero. Differentiable surface triangulation. *ACM Transactions on Graphics*, 2021.

[41] Rohan Sawhney and Keenan Crane. Monte carlo geometry processing: A grid-free approach to PDE-based methods on volumetric domains. *ACM Transactions on Graphics*, 2020.

[42] Nicholas Sharp and Maks Ovsjanikov. PointTriNet: Learned triangulation of 3D point sets. In *Proc. European Conf. on Computer Vision*, 2020.

[43] Tianchang Shen, Jun Gao, Kangxue Yin, Ming-Yu Liu, and Sanja Fidler. Deep marching tetrahedra: a hybrid representation for high-resolution 3D shape synthesis. In *Advances in Neural Info. Proc. Systems*, 2021.

[44] Hang Si. TetGen, a Delaunay-based quality tetrahedral mesh generator. *ACM Trans. Math. Softw.*, 2015.

[45] Ty Trusty, Honglin Chen, and David I.W. Levin. The shape matching element method: Direct animation of curved surface models. *ACM Transactions on Graphics*, 2021.

[46] Shubham Tulsiani, Hao Su, Leonidas J Guibas, Alexei A Efros, and Jitendra Malik. Learning shape abstractions by assembling volumetric primitives. In *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 2017.

[47] Dirce Uesu, Louis Bavoil, Shachar Fleishman, Jason Shepherd, and Cláudio T Silva. Simplification of unstructured tetrahedral meshes by point sampling. In *Intl. Workshop on Volume Graphics*, 2005.

[48] Nanyang Wang, Yinda Zhang, Zhuwen Li, Yanwei Fu, Wei Liu, and Yu-Gang Jiang. Pixel2Mesh: Generating 3D mesh models from single RGB images. In *Proc. European Conf. on Computer Vision*, 2018.

[49] Yu Wang, Alec Jacobson, Jernej Barbič, and Ladislav Kavan. Linear subspace design for real-time shape deformation. *ACM Transactions on Graphics*, 2015.

[50] Yunhai Wang, Shmulik Asafi, Oliver Van Kaick, Hao Zhang, Daniel Cohen-Or, and Baoquan Chen. Active co-analysis of a set of shapes. *ACM Transactions on Graphics*, 2012.

[51] Jiajun Wu, Chengkai Zhang, Tianfan Xue, William T Freeman, and Joshua B Tenenbaum. Learning a probabilistic latent space of object shapes via 3D generative-adversarial modeling. In *Advances in Neural Info. Proc. Systems*, 2016.

[52] Kangxue Yin, Zhiqin Chen, Siddhartha Chaudhuri, Matthew Fisher, Vladimir G Kim, and Hao Zhang. COALESCE: Component assembly by learning to synthesize connections. In *Intl. Conf. on 3D Vision*, 2020.

[53] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 2017.