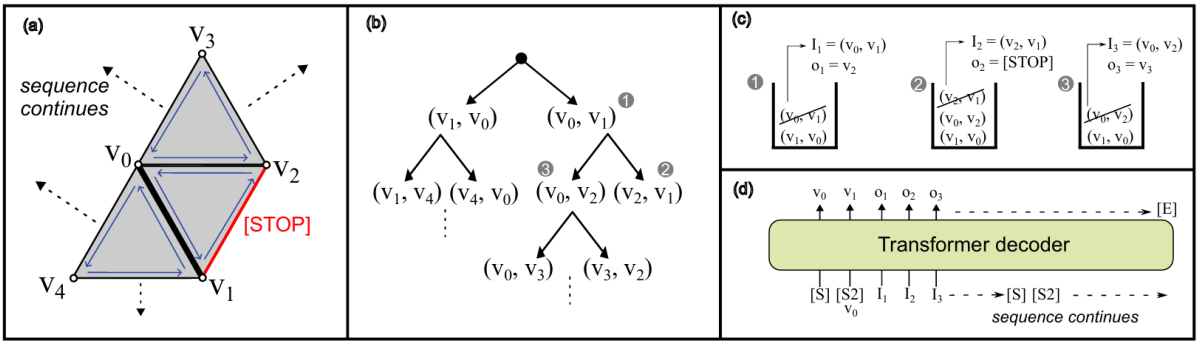
TreeMeshGPT: Artistic Mesh Generation with Autoregressive Tree Sequencing(CVPR 2025)

- **Author**
  - **TreeMeshGPT: Artistic Mesh Generation with Autoregressive Tree Sequencing**

    Stefan Lionar[1,2,3] Jiabin Liang[1,2,3] Gim Hee Lee[3]
    [1]Sea AI Lab [2]Garena [3]National University of Singapore
- **Background**
  - 在现有的3D生成技术中，voxel、point cloud、和implicit function经常被使用，在生成过程之后，像Marching Cubes这样的技术生成的网格过于稠密，不适合用于实时性渲染的场景。但是人工方法效果虽然好，但是费时费力，非常需要新的技术来弥补其中的空缺。
  - MeshAnything每个面使用9个token，产生的序列太长。时间复杂度为平方级别。MeshAnything V2 和EdgeRunner使用邻接的边来创造更短的序列以表示相同的网格，他们能使得网格在1600和4000面数之间，但是，现实世界需要更加准确的面数更多的网格来表示。
- **Contribution**
  - 提出了新的自回归数序列模型，可以有效地表示每个三角面上的两个标记。
  - 提出了7-bit标记化，能够生成高质量的网格，最多可达5500个面。
  - 大量实验表明，模型可以产生更高质量的网格并且可以产生真实世界的3D扫描。
- **Related Work**
  - **Mesh Extraction**
    - 在所有成功的方法中，Marching Cubes是最常被使用的。它将标量场划分为立方体，并提取三角形以近似等值面。
    - Dual Contouring和Dual Marching Cubes被使用来提高其能力。
    - Poisson reconstruction使用点云和法线作为边界条件来解决定义在3D空间中的标量场，然后应用轮廓立方体算法提取网格。
  - **3D Generation**
    - 最早的方法依赖于优化基础表示，以模拟来自二维图像或多视角二维图像的条件。
    - 随着大数据的兴起，前馈神经网络逐渐被使用，速度显著提升但是质量低。
- **Method**
  - **标记化**
    - 标记化涉及将顶点、边或面编码为模型可以逐步处理的顺序标记。

- 主要完成的工作为处理网格（将网格规格化）、在点云中取样、去除重复的边、准备半边网格（每条无向边拆分成为两条有向边）。

- **Sequence order**

  - 

  - **算法步骤**

    - 1. 初始化栈，栈中包括（V_0, V_1）和（V_1, V_0）。（将一个边分成两个有向边的过程，叫做half-edge）
    - 2. 将栈顶元素出栈（此例中为（V_0, V_1））。
    - 3. 找到此边（V_0, V_1）所对应的点（V_2），此过程中，按照逆时针方向查找下一个点，并将V_2记为o_1。如果在查找下个点时并未找到，则标记一个【STOP】标签。
    - 4. V_2所对应边（V_0, V_2）、（V_2, V_1）压入栈。

- **Generation process**

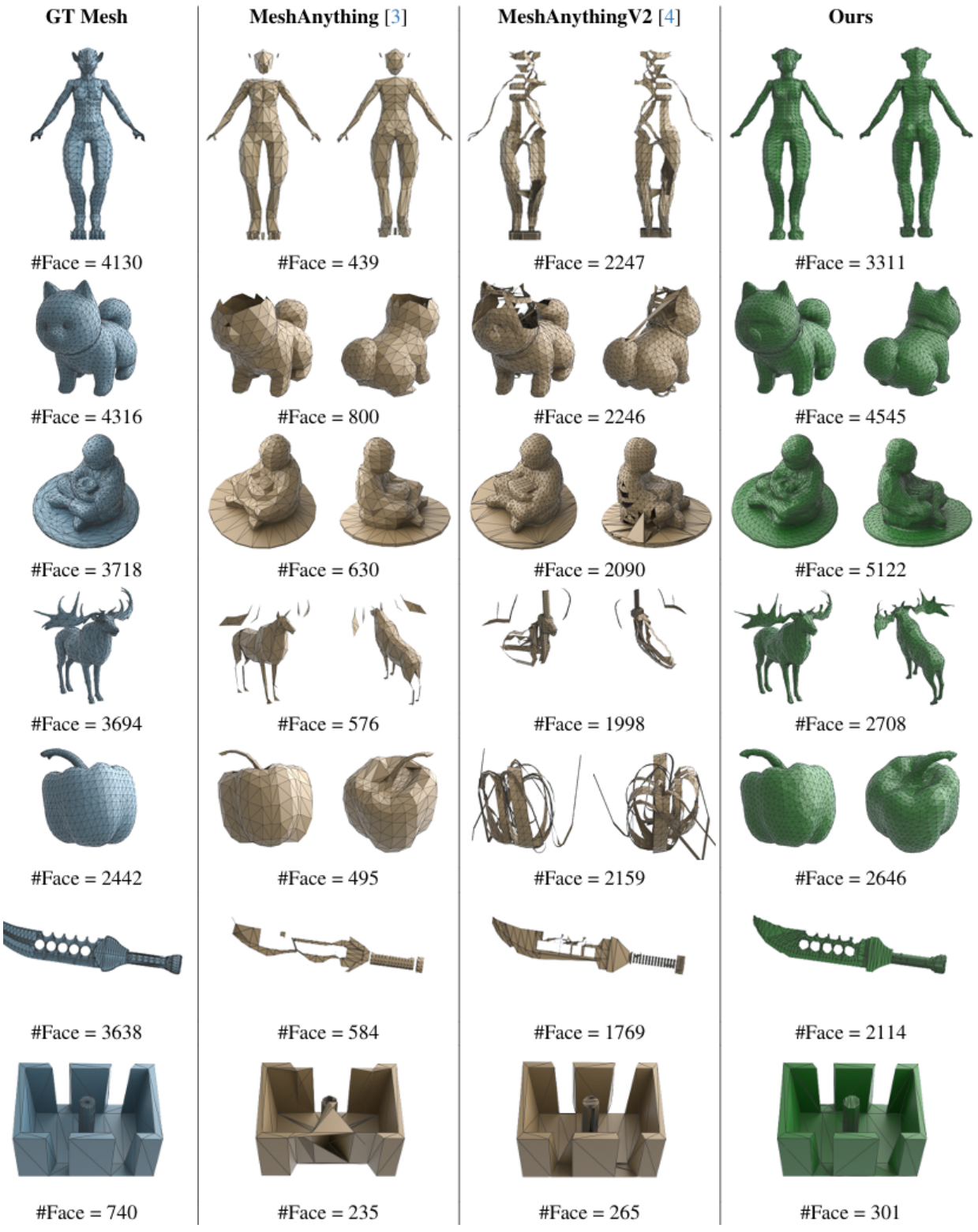
  - 初始化时【SOS】用来预测第一个顶点，【SOS2】用来预测第二个顶点，用此两点来构成自回归树的初始输入。
  - 当栈空时，则证明当前面已经完成，下一个面需要新的【SOS】、【SOS2】。
  - 在所有面完成以后，序列会以一个【EOS】标签结尾。
  - 其中
  - $$\mathcal{M} = \bigcup_{n=1}^{N} (v_1^n, v_2^n, v_3^n)$$
  - M代表整个网格
  - $$I_n \notin \{\texttt{[SOS]}, \texttt{[SOS2]}\}$$
  - I_n代表输入，[SOS] 和 [SOS2] 这两个特殊 token 只是用来“启动”一个连通分量的两条初始边，它们本身并不对应任何真实存在于最终网格里的边，也不会触发三角形生成。
  - $$o_n \notin \{[\text{STOP}], [\text{EOS}]\}$$
  - 当 o⊠ = [STOP] 或 o⊠ = [EOS] 时，模型不会输出一个具体的顶点坐标，而是输出一个“终止”信号。

- **Input embedding**
  - 引用其他论文内容（《A 3d shape representation for neural fields and generative diffusion models.》），将R^3-->R^C，即将3d坐标转化为C维embedding。
  - TreeMeshGPT的输入嵌入模块负责把“一条有向边”转化成Transformer可以处理的向量。
  - 先将3D位置编码函数PosEmbed(.)把两个端点坐标（V_i, V_j），映射到各自的C维向量，得到PosEmbed(V_i)、PosEmbed(V_j)。再将这两个C维向量拼接成为2C维向量，最后通过一个MLP将2C维向量映射到Transformer的隐藏维度D，得到最终的输入。
- **Vertex prediction**
  - 使用了一个MLP heads。
- **Advantages**
  - 与普通的标记化方法相比，这种高效的排序方法在大多数网格中实现了大约22%的压缩率，接近MeshAnything V2和EdgeRunner的两倍，并且，通过使用动态栈的方法，使得预测下一步的效率更加提升，而且能够最大限度的解决反转法线的问题。
- **Loss function**
  - $$\mathcal{L} = \mathcal{L}_{\text{CE}}(\mathbf{O}_x, \hat{\mathbf{O}}_x) + \mathcal{L}_{\text{CE}}(\mathbf{O}_y, \hat{\mathbf{O}}_y) + \mathcal{L}_{\text{CE}}(\mathbf{O}_z, \hat{\mathbf{O}}_z)$$

Experiments

- **Dataset**
  - 使用了Objaverse meshes作为数据集，并且为了保证网格质量足够，论文使用了尽量能够满足半边遍历的要求，如他们都是流形并且没有反转法线。
  - 所有的网格都被中心化和标准化处理。
  - 使用7-bit离散化，删除重复的面，并且选择面数少于5500的网格。
  - 每一个网格都会被在一个[0.75, 0.95]范围内进行缩放。
  - 在x轴和y轴按照30%的概率进行90^o ~ -90^o的旋转，在z轴按照均匀的方式从180^o ~ -180^o之间旋转。
- **Result**
  - **Qualitative comparison on Objaverse dataset**
    - **Quantitative results on Objaverse evaluation set**
Model	CD↓	NC↑	|NC| ↑
MeshAnything [3]	0.0115	0.223	0.853
MeshAnythingV2 [4]	0.0102	0.167	0.843
Ours	**0.0070**	**0.798**	**0.880**
    - **Quantitative results on our controlled experiment**
      -

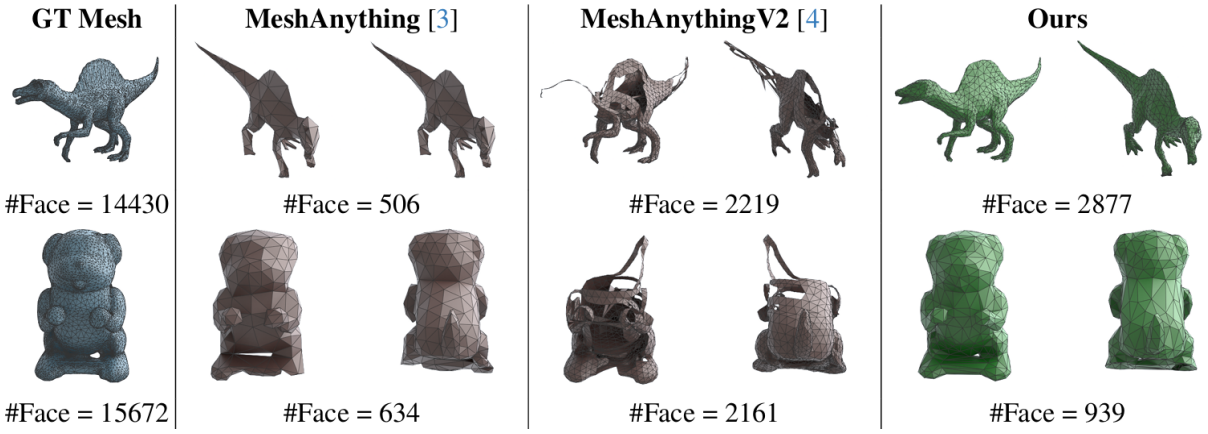
Tokenizer	CD↓	NC↑	|NC| ↑	Sequence Length↓
Naive [2]	0.0376	0.639	0.822	$9N_f$
VQ-VAE [33]	0.0352	0.673	0.815	$6N_f$
AMT [4]	0.0327	-0.069	0.768	$\pm 4N_f$
Ours	**0.0100**	**0.734**	**0.874**	$\mathbf{2N_f + 2N_c}$



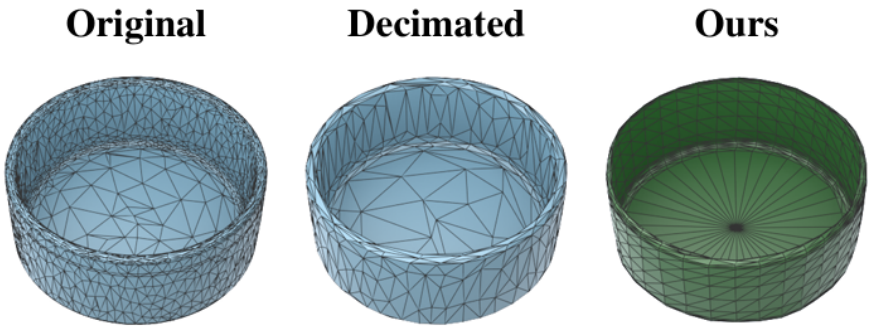
- Qualitative comparison on GSO dataset
  - Quantitative results on GSO dataset
    -

Model	CD↓	NC↑	|NC| ↑
MeshAnything [3]	0.0105	0.453	0.869
MeshAnythingV2 [4]	0.0116	0.3269	0.865
Ours	**0.0077**	**0.842**	**0.897**

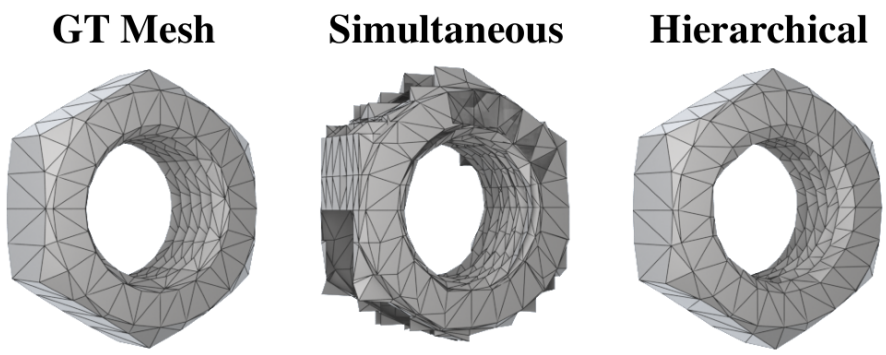
    -



  - Comparison between the decimated mesh and our output
    -



- Albation Study
  - MLP head ablation
    -



- Training perplexity comparison Between BFS and DFS traversals
  -