NeuFrameQ: Neural Frame Fields for Scalable and Generalizable Anisotropic Quadrangulation
Supplementary Material

A. Dataset Preparation

The original data are from open source 3D datasets [1, 2]. As they only provide triangulated meshes, we get the wireframe metadata from the model original sources. Then we merge pairs of triangle faces to form polygon meshes and keep samples with quad faces comprising more than 90% of total faces. This results in around 200k polygon meshes. We then expand the dataset with splitting components and keep mesh components containing more than 10% of the total face count. In this way, we expand the initial dataset to a size of 400k.

To improve the quality of the dataset, we apply several filters to remove those that are potentially not plausible for predicting frame fields. We consider three indicators on all meshes: the total surface area, the number of faces, and the number of vertices, and remove meshes where one of those indicators lies outside the range of [5%,95%] percentiles. This filtering removes, for example, many quadrangulated scanned objects and oversimplified meshes. Then, we exclude models that do not have a high proportion of vertices of degree 4 as we find that there exist some "fake" quad meshes simply created by subdividing a triangular mesh. Finally we get around 270k meshes.

B. Frame Field Calculation

The ground truth frame fields [6] are calculated similarly to Dielen et al. [3]. For each sample point p, we locate the quad face that contains p and interpolate the vectors of the opposite edges by the distance of p to both edges to calculate one of the representative vectors. Formally, as shown in Fig. 1, the frame is $u = \frac{s_2}{s_0+s_2}e_0 - \frac{s_0}{s_0+s_2}e_2, v = \frac{s_3}{s_1+s_3}e_1 - \frac{s_1}{s_1+s_3}e_3$.

We sample 10,000 meshes and calculate the frames on 40,000 sampled points per mesh and show the statistics in Fig. 2. For each mesh, we divide the frames by the average of the cross product of the two components of each frame, which corresponds to the average face area of the target quad mesh, to exclude the global scale. The resulting magnitudes are close to a log-normal distribution. Since data with Gaussian-like distributions are preferable for diffusion model training, we use the logarithmic magnitude as the denoising target. During inference, we recover the original scale via exponentiation.

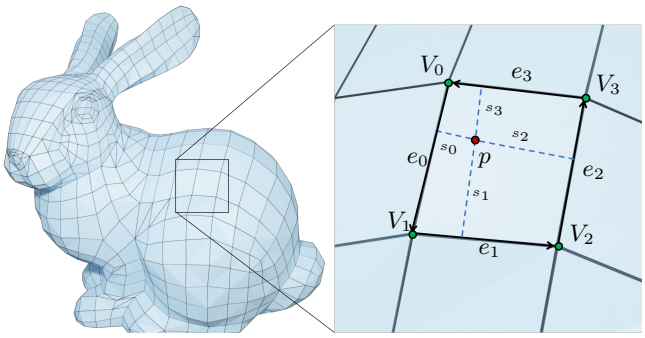

Figure 1. **The elements of frame field calculation.** The frame at each point is determined by the located quadrilateral face. The representative vectors are calculated as a weighted summation of the directed opposite sides.

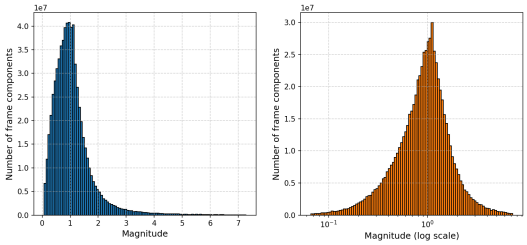

Figure 2. The distribution of frame magnitudes from sampled 10,000 meshes.

C. Network Details

We make several modifications to the original PTv3 [7] network to make it suitable for our tasks. We add a linear layer as a residual path across the voxelization operator to make points in the same voxel distinguishable between each other.

In Tab. 1, we list the specific model configurations of our direction regression network and magnitude diffusion

network, which both use PTv3 [7] as the backbone. Besides the backbone configuration, in the magnitude diffusion model, we also have the attention-based encoder and decoder. Both the encoder and decoder consist of 2 self attention blocks. For the magnitude diffusion network, we remove the randomness, i.e. dropout and random shuffle orders, to improve consistency at each timesteps. We found this modification improves the performance for the diffusion process.

Both our direction regression network and magnitude denoising networks are trained on the dataset described above. For direction regression, we use AdamW [5] optimizer with a batch size of 288 meshes and schedule the learning rate as a cosine decay following a linear warm-up with $lr_{max} = 5 \times 10^{-4}$. For the magnitude diffusion network, we use AdamW [5] optimizer with a batch size of 96 meshes and schedule the learning rate as a constant value of 10^{-4}. We use the cosine scheduler for α_t and rescale it to achieve zero SNR [4] at $t = T$. We augment each mesh with random shifts and random rotations around the gravity axis and then uniformly sample a point cloud of $50,000 \sim 60,000$ surface points as training data to capture geometry details as much as possible.

D. More Results

We show more results in Figs. 3 and 4. To explicitly demonstrate the ability to process non-manifold inputs, Fig. 5 shows a remeshed result from a non-manifold ShapeNet object.

References

[1] Matt Deitke, Ruoshi Liu, Matthew Wallingford, Huong Ngo, Oscar Michel, Aditya Kusupati, Alan Fan, Christian Laforte, Vikram Voleti, Samir Yitzhak Gadre, Eli VanderBilt, Aniruddha Kembhavi, Carl Vondrick, Georgia Gkioxari, Kiana Ehsani, Ludwig Schmidt, and Ali Farhadi. Objaverse-xl: A universe of 10m+ 3d objects. In *NeurIPS*, 2023. 1

[2] Matt Deitke, Dustin Schwenk, Jordi Salvador, Luca Weihs, Oscar Michel, Eli VanderBilt, Ludwig Schmidt, Kiana Ehsani, Aniruddha Kembhavi, and Ali Farhadi. Objaverse: A universe of annotated 3d objects. In *CVPR*, pages 13142–13153. IEEE, 2023. 1

[3] Alexander Dielen, Isaak Lim, Max Lyon, and Leif Kobbelt. Learning Direction Fields for Quad Mesh Generation. *Computer Graphics Forum*, 40(5):181–191, 2021. 1

[4] Shanchuan Lin, Bingchen Liu, Jiashi Li, and Xiao Yang. Common diffusion noise schedules and sample steps are flawed. In *WACV*, pages 5392–5399. IEEE, 2024. 2

[5] I Loshchilov. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017. 2

[6] Daniele Panozzo, Enrico Puppo, Marco Tarini, and Olga Sorkine-Hornung. Frame fields: Anisotropic and non-orthogonal cross fields. *ACM Transactions on Graphics*, 33 (4):1–11, 2014. 1

[7] Xiaoyang Wu, Li Jiang, Peng-Shuai Wang, Zhijian Liu, Xihui Liu, Yu Qiao, Wanli Ouyang, Tong He, and Hengshuang Zhao. Point transformer V3: simpler, faster, stronger. In *CVPR*, pages 4840–4851. IEEE, 2024. 1, 2

Network	Config	Value
Direction Regression	serialization pattern	Z + TZ + H + TH
	shuffle orders	True
	encoder depth	[2, 2, 2, 6, 2]
	encoder channels	[64, 64, 128, 256, 512]
	encoder num heads	[4, 4, 8, 16, 32]
	encoder patch size	[1024, 1024, 1024, 1024, 1024]
	decoder depth	[2, 2, 2, 2]
	decoder channels	[64, 64, 128, 256]
	decoder num heads	[4, 4, 8, 16]
	decoder patch size	[1024, 1024, 1024, 1024]
	down stride	[$\times$ 2, $\times$ 2, $\times$ 2, $\times$ 2]
	mlp ratio	4
	qkv bias	True
	drop path	0.3
Magnitude Diffusion	serialization pattern	Z + TZ + H + TH
	shuffle orders	False
	encoder depth	[2, 2, 2, 6, 2]
	encoder channels	[128, 128, 128, 256, 512]
	encoder num heads	[8, 8, 8, 16, 32]
	encoder patch size	[1024, 1024, 1024, 1024, 1024]
	decoder depth	[2, 2, 2, 2]
	decoder channels	[64, 64, 128, 256]
	decoder num heads	[4, 4, 8, 16]
	decoder patch size	[1024, 1024, 1024, 1024]
	down stride	[$\times$ 2, $\times$ 2, $\times$ 2, $\times$ 2]
	mlp ratio	4
	qkv bias	True
	drop path	0.0
	encoder channels	128
	encoder num attention layers	2
	encoder num heads	8
	decoder channels	64
	decoder num attention layers	2
	decoder num heads	4
	timestep embedding channels	512

Table 1. **Model Configurations**. For the serialization pattern, Z is Z-order, TZ is Trans Z-order, H is Hilbert, TH is Trans Hilbert.

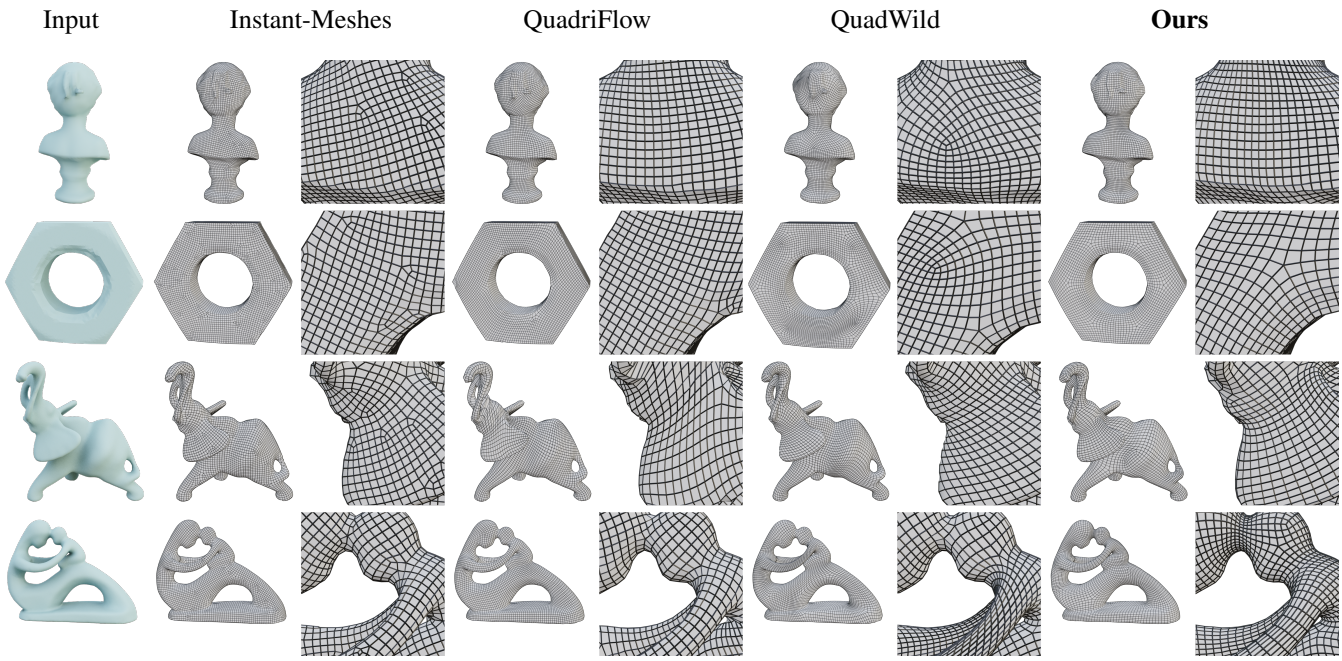

Figure 3. More qualitative comparison results with other mesh quadrangulation methods.

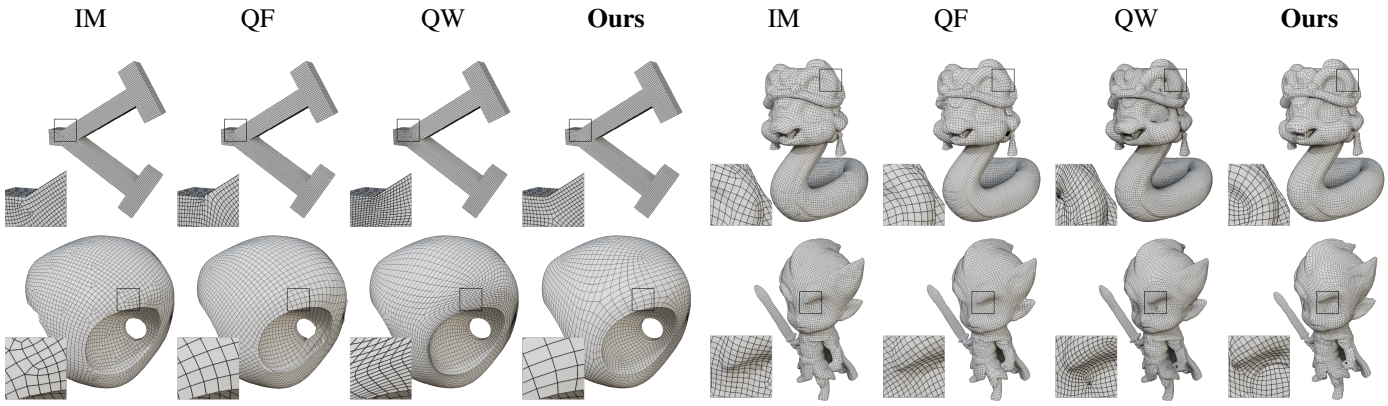

Figure 4. More qualitative comparisons on diverse mesh sources. (left: Thingi10k, right: 3D generation tools).

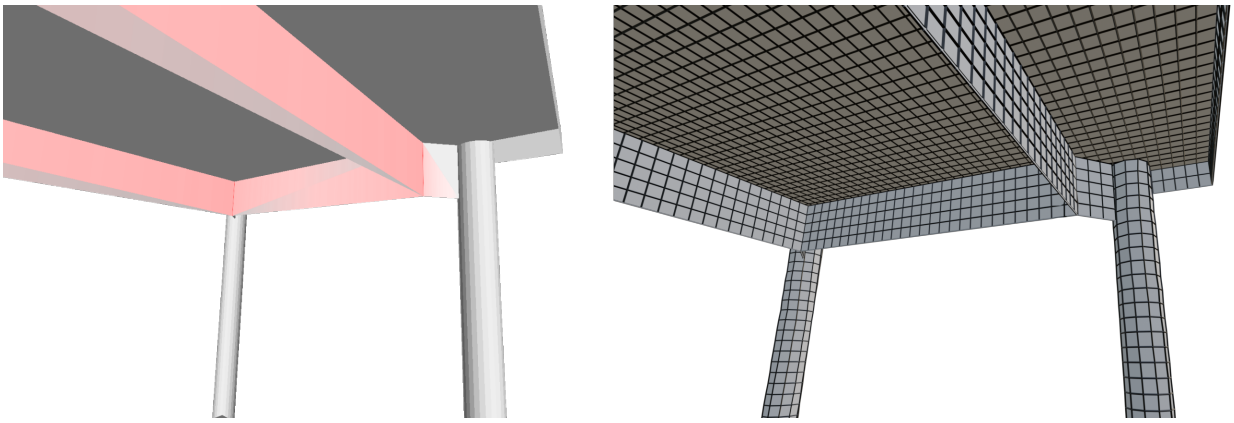

Figure 5. Non-manifold mesh input and the remeshed version. Non-manifold edges and their adjacent faces in the input are highlighted in red.