# Hyper-MD: Mesh Denoising with Customized Parameters Aware of Noise Intensity and Geometric Characteristics

Supplementary Material

## 1. Overview

We propose a hyper-network-based MD method called Hyper-MD, which effectively denoises meshes by utilizing customized parameters that are aware of noise intensity and geometric characteristics of each facet. In this supplementary material, we present additional details and discussions regarding the proposed method. The following sections are included to provide a comprehensive understanding of Hyper-MD:

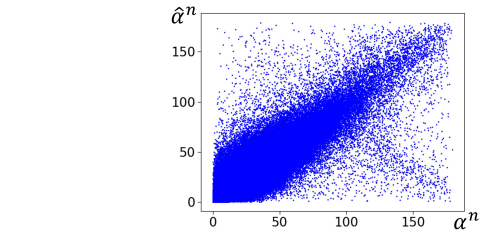
- Studies on the Agent.
- Implementation Details.
- Studies on Customized Layers.
- Studies on Hyperparameters.
- Additional Experimental Results

## 2. Studies on the Agent

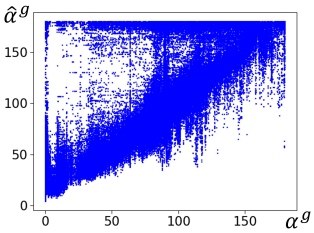
As described in Subsection **3.2**, denoising is performed without access to clean meshes. To address this issue, we develop an agent that approximates clean meshes through filtering, enabling the calculation of $\alpha^n$ and $\alpha^g$. In this section, we discuss the rationale behind the agent's design. We generate $\hat{\alpha}^n$ and $\hat{\alpha}^g$ based on clean meshes, and plot the relationships between various angles in Figure 1. Subplot (a) illustrates a clear positive correlation between $\alpha^n$ and $\hat{\alpha}^n$. This indicates that the agent's estimation of $\alpha^n$ aligns well with the true values derived from clean meshes. Furthermore, subplot (b) depicts the relationship between $\alpha^g$ and $\hat{\alpha}^g$, while subplot (c) represents the relationship between $\alpha'^g$ (calculated based on the original noisy meshes) and $\hat{\alpha}^g$. Comparing subplot (b) and subplot (c), we observe that the correlation between $\alpha^g$ and $\hat{\alpha}^g$ is more pronounced. In summary, the filtered mesh serves as a reasonable agent of the clean mesh, as evidenced by the clear correlations between the estimated angles and their respective ground truth values. This justifies the use of the filtered mesh as an agent for the clean mesh in our approach.

## 3. Implementation Details

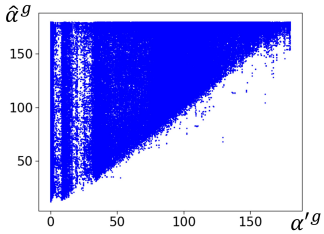
This section provides implementation details of the method. The facet attributes utilized in Hyper-MD include the normal, centroid, and area. During the training of Hyper-MD, we initialize the weights using a truncated normal distribution. The employed optimizer is Adam with default parameter settings in PyTorch, namely $\beta_1 = 0.9$, $\beta_2 = 0.9$, and $\epsilon = 10^{-8}$. We set the batch size to 512 for training. The choice of batch size may vary depending on the specific




(a) Relationship between $\hat{\alpha}^n$ and $\alpha^n$.




(b) Relationship between $\hat{\alpha}^g$ and $\alpha^g$.




(c) Relationship between $\hat{\alpha}^g$ and $\alpha'^g$.

Figure 1. Relationship between angles calculated using different meshes. $\alpha^g$ and $\alpha^n$ are obtained based on filtered meshes. $\hat{\alpha}^g$ and $\hat{\alpha}^n$ are calculated based on clean meshes. $\alpha'^g$ is computed on noisy meshes.

dataset and computational resources available. The training of Hyper-MD is conducted on a computer equipped with an AMD Ryzen 9 5900HX CPU and a NVIDIA GeForce RTX 3080 Laptop GPU.

On the SynData dataset, we randomly select 600 faces from each noisy mesh to participate in training during every epoch. In the first training step, the denoising network is trained for 1000 epochs. The learning rate is initially set to 0.0001 and is halved after the 600th and 800th epochs. In the second training step, each denoising model is trained for 500 epochs. The learning rate is set to 0.00001, which is lower than the initial learning rate used in the first training step. In the third training step, the hyper-network is trained for 500 epochs. The learning rate remains at 0.00001. This step focuses on training the hyper-network to dynamically customize network parameters for each facet, enhancing the overall denoising performance.

Regarding the three Kinect datasets, it is worth noting that the noise intensity across all noisy meshes is similar. As a result, there is no need to consider noise intensity as a factor, and therefore, $\alpha^n$ is no longer input into the hyper-network. Furthermore, when dealing with the Kinect datasets, there are only four candidates for coarse-grained bases. These bases correspond to four categories

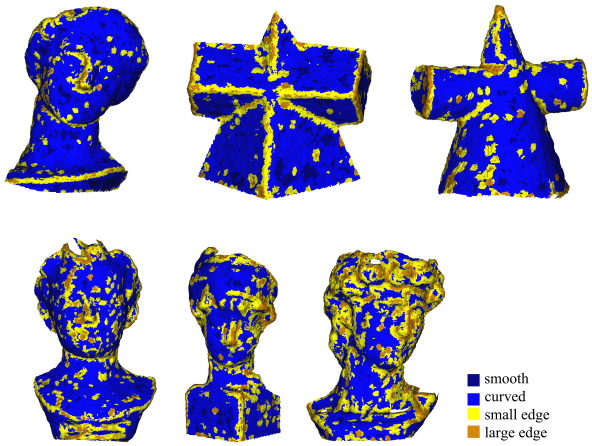



Figure 2. The schematic diagram of different facet categories.

Table 1. The experimental results for studies on customized layers. ✓means that the corresponding fully connected layer is customized by the hyper-network.

| Variants | $fc_1$ | $fc_2$ | $fc_3$ | $fc_4$ | Simple | | Complex | |
|---|---|---|---|---|---|---|---|---|
| | | | | | $E_a$ | $E_v$ | $E_a$ | $E_v$ |
| $V_1$ | × | × | × | ✓ | 4.93 | 2.61 | 5.87 | 10.44 |
| $V_2$ | × | × | ✓ | ✓ | 4.84 | 2.49 | 5.60 | 9.11 |
| $V_3$ | × | ✓ | ✓ | ✓ | 4.45 | 2.33 | 4.68 | 8.56 |
| Best | ✓ | ✓ | ✓ | ✓ | **4.32** | **2.20** | **4.50** | **7.85** |

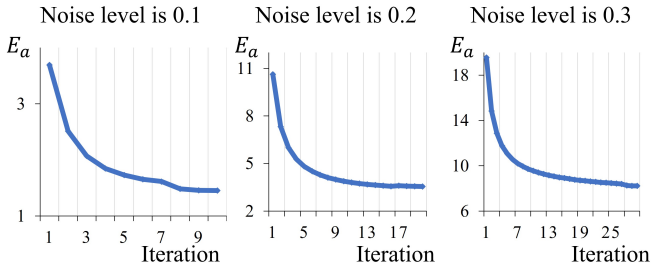



Figure 3. $E_a$ decreases as the iteration number increases.

that are divided based on $\alpha^g$, as depicted in the schematic diagram shown in Figure 2. During the training process, we randomly select 200 facets from each noisy mesh in every epoch. In the first training step, the denoising network is trained for 500 epochs, starting with a learning rate of 0.0001. The learning rate is then halved after the 300th and 400th epochs. Moving on to the second training step, each denoising model is trained for 200 epochs, with a fixed learning rate of 0.00001. Finally, in the third training step, the hyper-network is trained for 200 epochs, also with a learning rate of 0.00001.

Table 2. The results of hyper-parameter selection experiments. $r_1$ denotes the number of rings for the input patch. $r_2$ is the number of rings for the patch used to calculate $\alpha^g$. $I$ is the iteration number of mean filtering. $n_c$ is the category number of facets.

| Settings | $r_1$ | $r_2$ | $I$ | $n_c$ | Simple | | Complex | |
|---|---|---|---|---|---|---|---|---|
| | | | | | $E_a$ | $E_v$ | $E_a$ | $E_v$ |
| Best | 3 | 2 | 20 | 4 | 4.32 | 2.20 | 4.50 | 7.85 |
| $V_1$ | **2** | 2 | 20 | 4 | 5.31 | 2.86 | 5.99 | 10.45 |
| $V_2$ | **4** | 2 | 20 | 4 | 4.92 | 2.31 | 5.32 | 9.51 |
| $V_3$ | 3 | **1** | 20 | 4 | 4.99 | 2.72 | 5.45 | 8.89 |
| $V_4$ | 3 | **3** | 20 | 4 | 4.84 | 2.67 | 5.22 | 8.46 |
| $V_5$ | 3 | 2 | **10** | 4 | 5.12 | 2.64 | 5.23 | 8.48 |
| $V_6$ | 3 | 2 | **30** | 4 | 4.86 | 2.54 | 5.31 | 8.39 |
| $V_7$ | 3 | 2 | 20 | **3** | 4.43 | 2.31 | 4.71 | 8.12 |
| $V_8$ | 3 | 2 | 20 | **5** | 4.31 | 2.23 | 4.56 | 7.97 |

## 4. Studies on Customized Layers

In Hyper-MD, the parameters of the four fully connected layers in $\mathcal{M}_{NI}$ are customized by the hyper-network. We have also conducted experiments where we customized partial layers, and the results are presented in Table 1. As expected, the performance improves as we customize more parameters. This is due to the higher flexibility that comes with customizing a greater number of parameters.

| Metrics | Clean | Noisy | Hyper-MD |
|---|---|---|---|
| Vertex Accuracy | 84.73% | 79.56% | 83.84% |
| Face Accuracy | 84.70% | 79.19% | 83.81% |

Table 3. The results of the human body segmentation task.

## 5. Studies on Hyper-parameters

In Hyper-MD, careful consideration is given to the design of several hyper-parameters, including the patch size of the input for the denoising network, the patch size for angle calculation, the iteration number of the mean filter, and the category number. These parameters are introduced and discussed in detail in Section **3. Methodology**. To evaluate the impact of these hyper-parameters, we conduct experiments on the SynData dataset and present the results in Table 2. Each row in the table corresponds to a specific parameter setting, with the first row representing the best parameter setting obtained. To facilitate the selection of each parameter, we adopt a systematic approach where each setting only modifies one parameter at a time. The modified parameter in each setting is clearly marked in bold, allowing for easy

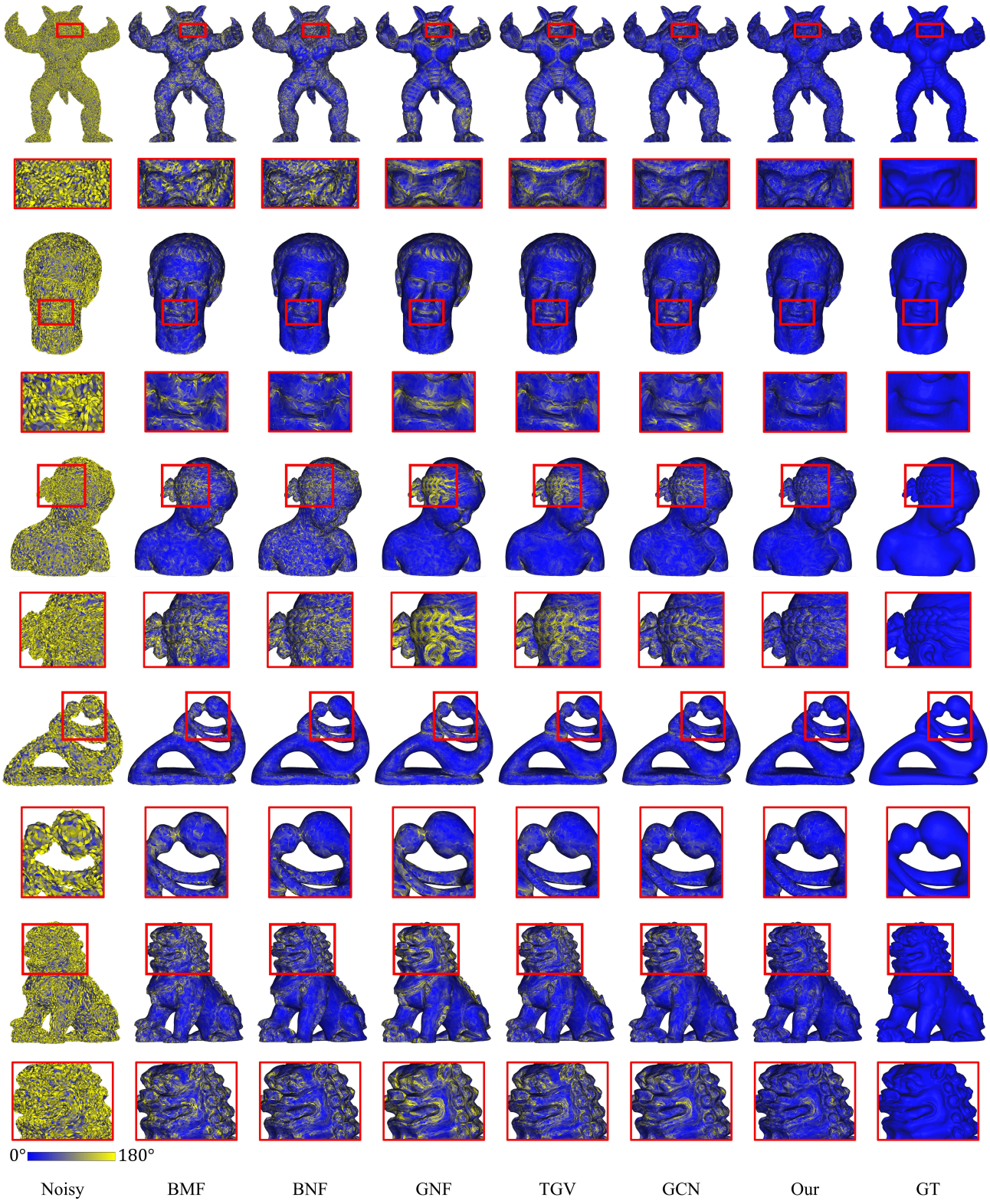



Figure 4. The results of complex shapes on SynData.

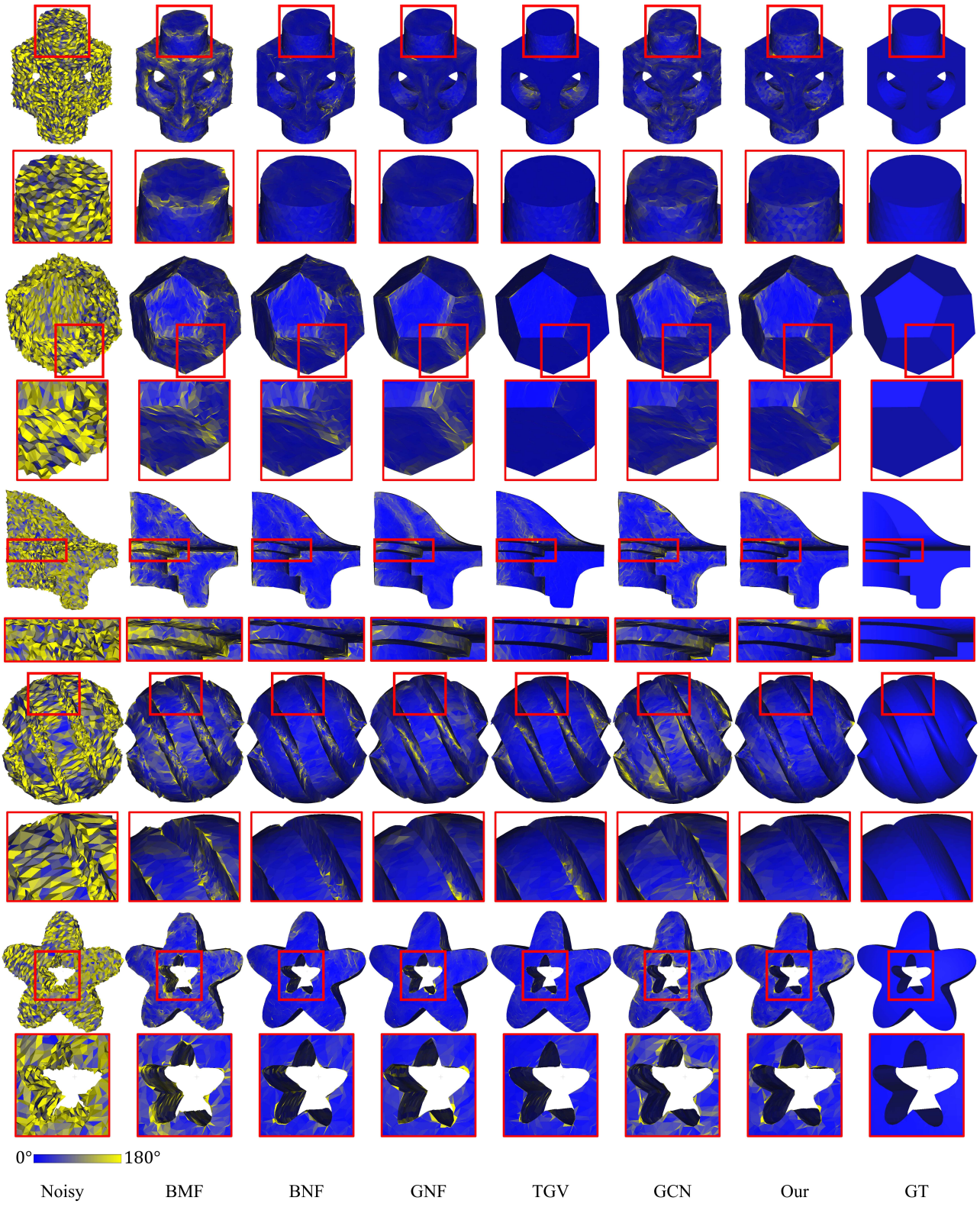



Figure 5. The results of simple shapes on SynData.

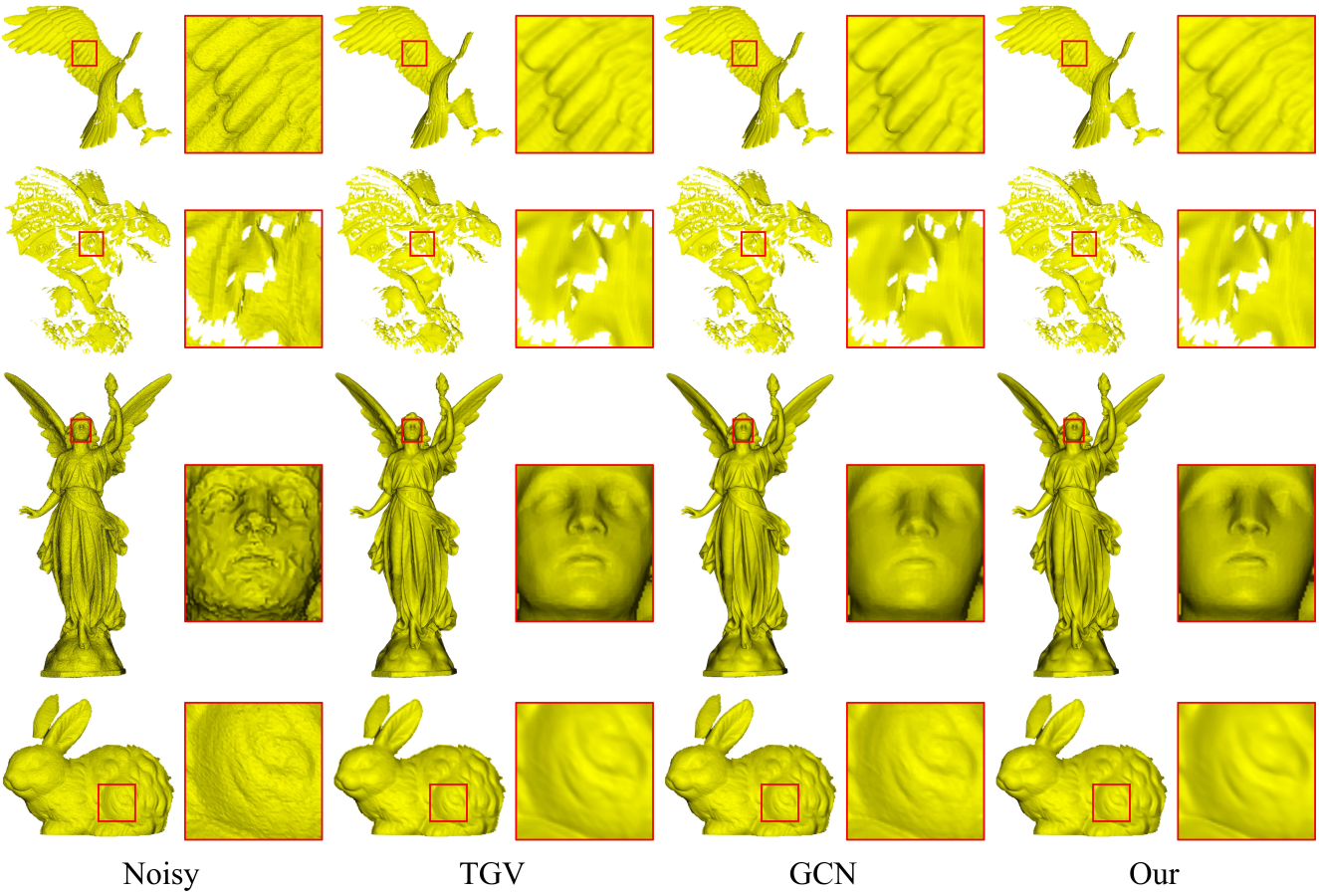



Figure 6. The denoising results of models collected from the internet.

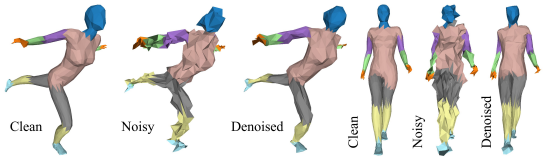



Figure 7. The results of the human body segmentation task.

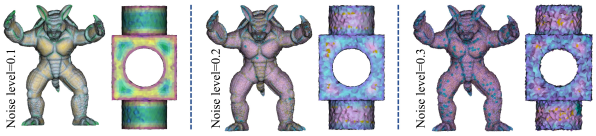



Figure 8. The color mapping of the offsets for each face.

identification and comparison. In addition to the parameters in Table 2, the $k$ for developing an agent is set to 20 on SynData, whereas on real-scanned datasets, $k$ equals 5.

Besides, we conduct some discussion on the category number of facets. The category number is determined by two factors. The first factor is noise intensity which is revealed by $\alpha^n$. In the Synthetic dataset, we have introduced three levels of noise to clean meshes, resulting in the natural division of facets into three categories based on $\alpha^n$. However, in the three Kinect datasets, where the noise level is consistent in each dataset, facets are not classified based on noise intensity. The second factor is the geometric characteristic, described by $\alpha^g$. Following the approach in [3] and [2], we classify facets into 4 categories with different geometric characteristics. Nevertheless, we still try 3 and 5 categories, and collect the experimental results in Table 2, from which we can see that 4 categories perform best.

## 6. Additional Experimental Results

Due to the limitation of paper length, some experimental results cannot be included in the main body. This section provides additional results.

In our experiments, we fixed the iteration numbers at 10, 20, and 30 for noise levels of 0.1, 0.2, and 0.3, respectively. The results of different iteration numbers are depicted in Figure 3. We can observe that as the iteration number in-

creases, the angle error ($E_a$) decreases, indicating improved denoising performance. It is important to note that further iterations could potentially optimize the denoising output even more. However, due to the absence of clean meshes during the denoising process, exhaustive search for the optimal iteration number is not feasible.

Figure 4 shows the denoising results of all complex shapes on the SynData dataset, while Figure 5 presents the results of simple shapes. In both cases, the noise level is set to 0.3. We can see that our results show the least yellow and the most blue In Figure 4, indicating superior denoising performance. However, in Figure 5, TGV outperforms the proposed Hyper-MD.

Figure 6 shows the denoising results of models collected from Internet. We can see that the performance of the compared methods is similar, except that our method produces better feature recovery results, which further verifies the capability of Hyper-MD.

To verify the efficacy of the proposed mesh denoising method, we conduct experiments on a human body segmentation task using Laplacian2Mesh [1]. The results in Table 3 and Figure 7 demonstrate that our method provides non-trivial enhancements.

For each face, we compress its $\mathbf{W}_4^f$ into a 3D vector using PCA, and then scale it to the range of 0-255, representing the RGB values of the face. Some colored meshes are shown in Figure 8. As expected, the offsets are related to geometry and noise.

## References


[1] Qiujie Dong, Zixiong Wang, Manyi Li, Junjie Gao, Shuangmin Chen, Zhenyu Shu, Shiqing Xin, Changhe Tu, and Wenping Wang. Laplacian2Mesh: Laplacian-based mesh understanding. *IEEE Transactions on Visualization and Computer Graphics*, 2023. 6

[2] Yuefan Shen, Hongbo Fu, Zhongshuo Du, Xiang Chen, Evgeny Burnaev, Denis Zorin, Kun Zhou, and Youyi Zheng. GCN-Denoiser: Mesh denoising with graph convolutional networks. *ACM Transactions on Graphics (TOG)*, 41(1):1–14, 2022. 5

[3] Wenbo Zhao, Xianming Liu, Yongsen Zhao, Xiaopeng Fan, and Debin Zhao. NormalNet: Learning-based mesh normal denoising via local partition normalization. *IEEE Transactions on Circuits and Systems for Video Technology*, 31(12): 4697–4710, 2021. 5