Towards Detailed 3D Modeling: Mesh Super-Resolution via Deformation

Ryo Tamura[†‡], Seiya Ito[†‡], Naoshi Kaneko[*§], Kazuhiko Sumi[*§]
[*]Department of Integrated Information Technology, Aoyama Gakuin University, Kanagawa, Japan.
[†]Graduate School of Science and Engineering, Aoyama Gakuin University, Kanagawa, Japan.
[‡]{tamura.ryo, ito.seiya}@vss.it.aoyama.ac.jp, [§]{kaneko, sumi}@it.aoyama.ac.jp

***Abstract*—This paper presents a method for constructing a detailed mesh model by deforming a coarse mesh model using surface normals estimated from images. Mesh representation is a 3D data structure that can represent a detailed shape with fewer parameters compared with other data structures. Multi-view stereo (MVS) algorithms are suitable for reconstructing a detailed mesh model from images captured from different viewpoints; however, the computation time increases rapidly as the resolution and the number of images increase. To overcome this problem, we propose a novel approach to deform a coarse mesh model. We first reconstruct a coarse mesh model by applying the Delaunay triangulation to a coarse point cloud obtained through structure-from-motion. The resulting model is deformed based on the changes in the surface normals estimated from images. In our experiments, we demonstrate that our method constructs detailed 3D models faster than the MVS with a single model dataset and general scene dataset.**

***Contribution*—Mesh deformation based on the normal maps generates a detailed mesh model.**

***Keywords*—3D reconstruction; 3D mesh model; super-resolution**

I. Introduction

One of the challenges in computer vision is modeling 3D scenes from multi-view images. Accurate and detailed models are used in a wide range of applications such as robotics and augmented reality.

One of the typical image-based 3D reconstruction approaches is the structure-from-motion (SfM). SfM extracts keypoints from images and recovers 3D coordinates by matching them with the images. The 3D model is represented as a point cloud, which is a collection of 3D coordinates. Furthermore, SfM estimates camera models and poses of images and these are used for multi-view stereo (MVS) algorithms. SfM recovers a coarse point cloud in areas where the keypoints match, whereas MVS reconstructs dense point clouds using matched areas among images. Hence, MVS is suitable for reconstructing a detailed mesh model from multi-view images. However, the computation time of MVS increases rapidly as the resolution and number of images increase. To overcome these problems, we first consider the procedure of MVS reconstruction. The computation time increases owing to the need to verify the correspondence between image patches. Moreover, point clouds are redundant in representing a detailed model because several points are required to represent a simple plane. Therefore, we employ a mesh that is an efficient representation of a 3D data structure, and directly operate the mesh of the 3D model, instead of verifying the correspondence between image patches.

In this paper, we propose a method for constructing a detailed mesh model by deforming a coarse mesh model. We first reconstruct a coarse point cloud model using SfM and then apply the Delaunay triangulation to the model. The obtained model is represented as triangle meshes. Then, we estimate the surface normals from images. To associate each mesh and the surface normals, we project the model to each camera using camera parameters acquired through SfM. For each triangle mesh, we deform the mesh based on the changes in the surface normals. This process is iterated until the difference between the mesh normal and the estimated surface normal is small. We refer to this problem of constructing a detailed mesh model while preserving the geometry of the underlying 3D structure as mesh super-resolution.

II. Related work

3D Shape Representations. One of the typical representation is a voxel, which is a regular grid in 3D space. As the voxel data structure is a 3D extension of pixels, various approaches for image processing can be easily extended to 3D data [1], [2], [3]. However, it is unsuitable for representing detailed 3D shapes owing to a trade-off between resolution and memory consumption. Another representation is a point cloud, which is a set of 3D points. It is typically used for classification [4], [5] because sensors that can directly capture point cloud data are available. The memory consumption of a point cloud is more efficient than that of a voxel representation. The conventional [6], [7] method is to upsample point cloud data to increase the resolution of a 3D model. However, it may be difficult to manage point cloud data because of the unstructured data structure.

In recent years, polygonal mesh representation, which is a set of vertices and faces, has been widely used [8], [9]. It exhibits almost the same memory efficiency as a point cloud, and has a spatial relationship between vertices and faces. Because mesh representation can represent planes with fewer parameters compared with point clouds, it is suitable for representing detailed 3D shapes. In this study, we represent 3D

978-1-7281-9331-1/20/$31.00 ©2020 IEEE

models as a set of triangle meshes that are a simple form of polygonal meshes.

MVS Reconstruction. MVS is used to reconstruct 3D models from a collection of images with known camera models and poses. SfM is often used to acquire camera models and poses from unstructured images. SfM approaches can be categorized into incremental [10] and global methods [11]. Incremental SfM reconstructs a 3D map from two initial images and incrementally adds new images to update the map. In contrast, global SfM simultaneously estimates all the camera poses. Hybrid SfM, which offers the advantages of incremental SfM and global SfM, has been proposed [12].

Conventional MVS approaches focus on the method of selecting neighboring images [13], and measure the correlation between image patches [14]. Schönberger presented an MVS method (COLMAP) that uses photometric and geometric priors for pixelwise view selection to estimate depth maps and surface normals. Although COLMAP achieved state-of-the-art performance in various MVS benchmarks, it requires a significant amount of time to reconstruct 3D models. In this study, we use the incremental SfM [15] part of COLMAP to reconstruct 3D models, and estimate the camera models and poses.

Deep Learning for Mesh Representation. Recently, deep learning-based methods for mesh representation have been proposed [16], [17], [18]. In these methods, a mesh model is regarded as a graph, and three operations are performed: graph convolution, graph pooling, and graph unpooling. Wang et al. presented Pixel2Mesh [16], which reconstructs a 3D object model from a single image. Pixel2Mesh uses an ellipsoid as the initial shape and deforms it using image features. Wen et al. proposed Pixel2Mesh++ [17], which is the extension of Pixel2Mesh; it reconstructs a 3D object model from multi-view images. Both methods can only reconstruct the same topology as the initial mesh. Therefore, it is difficult to use them for general scenes where various objects are mixed.

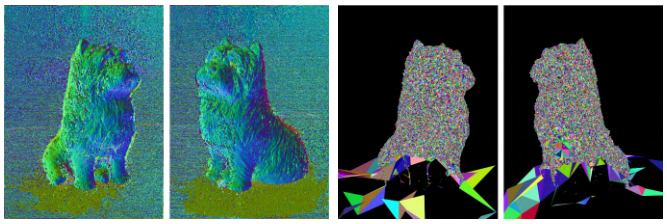
III. Proposed method

In this paper, we propose a novel method for constructing a detailed 3D model by deforming a coarse mesh model. The main objective is to reduce the computation time while maintaining the traditional MVS performance by directly deforming the mesh using surface normals. Fig. 1 shows an overview of the proposed system. We assume that an input coarse mesh model is constructed using multiple images captured from different viewpoints. Our method compares the surface normals of each mesh with those estimated from images, and then finds the changes in those surface normals.

First, we construct a coarse mesh model from multiple images. We utilize COLMAP as an SfM method to obtain a coarse point cloud model and the camera parameters of each viewpoint. The resulting coarse point cloud model is converted to a mesh model by applying the Delaunay triangulation algorithm. To associate each mesh of the mesh model and surface normals from the images, we project the mesh model to each viewpoint using standard computer graphics rendering pipeline and camera parameters acquired through SfM. Note that the normal map estimated from the images contains a surface normal at each pixel. In other words, the mesh itself contains only one surface normal, but the mesh contains multiple surface normals in image coordinates. We calculate the similarity of these surface normals to verify if the mesh should be deformed. The condition for deforming the mesh is whether the change in the surface normals exceeds the threshold value. We iterate the mesh deformation process until all the changes in the surface normals are small. The details of the proposed method are described as follows.

A. Normal Map Association

To construct a detailed mesh model, we use surface normals to deform a coarse mesh model. Although we estimate the surface normals from images, the resulting normal maps are not associated with the coarse mesh model. Therefore, we project each mesh of the coarse mesh model to the image coordinates using a standard computer graphics rendering pipeline. For rendering, camera intrinsic and extrinsic parameters are acquired through SfM and the image size is identical to the input image size. In this study, most meshes are associated with multiple pixels. As each pixel of the estimated normal maps has one surface normal and the mesh has only one surface normal, the changes in the estimated surface normal are candidates for deformation.



(a) Normal maps (b) Index maps

Fig. 2: Examples of normal maps and index maps.

B. Mesh Division

Mesh deformation is a two-step process: mesh division and mesh deformation. In the mesh division process, we divide one triangle mesh into two triangle meshes based on the surface normals, as shown in Fig. 2. This process is repeated using each estimated normal map sequentially until the difference between the surface normals of a mesh is small.

Fig. 3 shows an overview of the mesh division process. Let the target mesh of the mesh model be f, estimated normal map be N, and pixels where mesh f is projected be P. We denote the surface normal of mesh f as n_f and the estimated surface normal of pixel $i \in P$ as n_i. For the estimated normal map N, we calculate the angle θ_{ij} between the two surface normals n_i and n_j, where $i, j \in P, i < j$. If the maximum angle of θ_{ij} exceeds the threshold angle θ_ϵ, the target mesh f is divided. We assume that the normal changes linearly between two coordinates with the maximum angle. Because

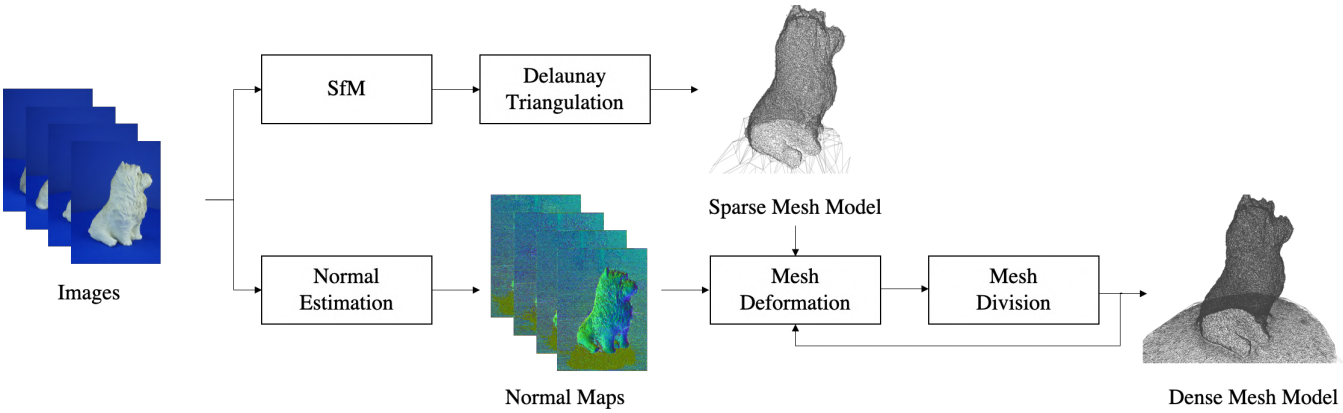


Fig. 1: **Overview of the proposed method.** The proposed method consists of three parts: normal estimation, mesh division, and mesh deformation.

the change in the normal is minimized by dividing the mesh at the midpoint, we calculate the midpoint p_m as follows:

$$p_m = (p_i + p_j)/2. \tag{1}$$

Here, p_i and p_j are the 3D coordinates of pixels i and j, respectively. Let v_k $(k = 1, 2, 3)$ be the vertices of mesh f. It is desirable to divide by a vertical straight line passing from the vertex to the midpoint. Hence, we calculate the angle of the lines from the midpoint p_m to p_i and v_k as follows:

$$cos\phi_{ik} = \frac{(p_i - p_m) \cdot (v_k - p_m)}{|(p_i - p_m)||(v_k - p_m)|}. \tag{2}$$

We divide the mesh by the line passing the midpoint p_m and the vertex $\tilde{v}$, which is the most perpendicular to the line passing through the coordinates of the two normals:

$$\tilde{v} = \arg\min_{v_k} |(cos\phi_{ik})|. \tag{3}$$

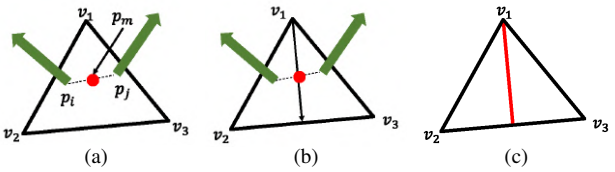


Fig. 3: **Mesh division overview.** The mesh division is divided into three parts: (a) calculating the midpoint, (b) calculating the position of the new edge, and (c) adding the new edge.

C. Mesh Deformation

In the deformation process, we use the dividing line passing through the vertex $\tilde{v}$ and the midpoint p_m. Let $\hat{v}$ be the intersection vertex of the dividing line and the edge and it is different from $\tilde{v}$. As illustrated in Fig. 4, we move the vertex $\hat{v}$ perpendicular to mesh f to minimize the difference between the two mesh normals and two estimated normals. The objective function is defined as follows:

$$F = \frac{n_i \cdot (p_i - \hat{v})}{|n_i||p_i - \hat{v}|} + \frac{n_j \cdot (p_j - \hat{v})}{|n_j||p_j - \hat{v}|}. \tag{4}$$

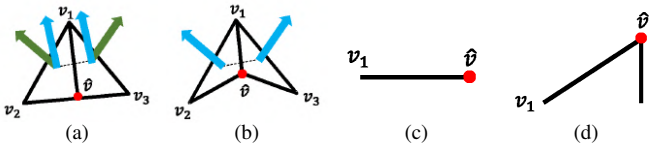


Fig. 4: **Mesh deformation overview.** The mesh deformation is divided into two parts: (a) calculating the movement of the normal, and (b) moving added point. (c) and (d) is a side view of (a) and (b), respectively.

IV. Experiments

In the experiments, we used the THU-MVS dataset [19] as a single object and the Benchmarking Camera Calibration 2008 [20] as an outdoor building. We used coarse mesh models generated by SfM as input models. Because these experiments aim to verify the effectiveness of mesh super-resolution, we used precise normal maps generated by MVS. In these experiments, we use the threshold $\theta_\epsilon = 10$ for the division and deformation using the normals.

We compared the three methods to confirm the effectiveness of the proposed method.

(1) MVS. We apply the Delaunay triangulation to the dense point cloud model generated by MVS. This was used as a baseline model.
(2) Uniform distribution. We added vertices with uniform distribution to mesh models.
(3) Proposed method. The proposed method used the coarse model as input.

A. Quantitative Evaluation

We compared the three models on the basis of the Earth mover's distance (EMD) and execution time. The EMD is

TABLE I: **Quantitative evaluation on the THU-MVS dataset.** Comparison between the 3D model generated by the proposed method and the 3D model generated by uniform distribution. The two models are evaluated quantitatively based on the mean and variance of the EMD. Using the model obtained by MVS, we measure the EMD for the two models. Furthermore, the execution time of the proposed method and the execution time of the MVS are measured and compared.

Method	# of Vertices	EMD (avg.)	EMD (var.)	Execution time [sec]
Dog RGB				
SfM (input)	3.381×10^3	-	-	-
Uniform distribution	1.008×10^4	1.763×10^{-2}	9.063×10^{-3}	-
Ours	1.220×10^5	$\mathbf{1.807 \times 10^{-3}}$	$\mathbf{6.248 \times 10^{-3}}$	$\mathbf{2.384 \times 10^2}$
MVS	1.870×10^5	-	-	5.490×10^5

TABLE II: **Quantitative evaluation on the Benchmarking Camera Calibration 2008.** Comparison between the 3D model generated by the proposed method and the 3D model generated by uniform distribution.

Method	# of Vertices	EMD (avg.)	EMD (var.)	Execution time [sec]
Herz-Jesus-P8				
SfM (input)	8.072×10^3	-	-	-
Uniform distribution	8.517×10^3	$\mathbf{2.541 \times 10^{-2}}$	$\mathbf{0.999 \times 10^{-2}}$	-
Ours	8.192×10^3	2.605×10^{-2}	1.049×10^{-2}	$\mathbf{2.628 \times 10^1}$
MVS	1.472×10^5	-	-	3.758×10^2
entry-P10				
SfM (input)	1.080×10^4	-	-	-
Uniform distribution	1.197×10^4	2.298×10^{-2}	$\mathbf{7.467 \times 10^{-3}}$	-
Ours	1.143×10^4	$\mathbf{2.404 \times 10^{-3}}$	7.555×10^{-3}	$\mathbf{8.937 \times 10^0}$
MVS	1.305×10^5	-	-	3.959×10^2
fountain-P11				
SfM (input)	1.380×10^4	-	-	-
Uniform distribution	2.156×10^4	$\mathbf{2.432 \times 10^{-2}}$	4.275×10^{-2}	-
Ours	2.105×10^4	2.548×10^{-2}	$\mathbf{3.322 \times 10^{-2}}$	$\mathbf{4.256 \times 10^0}$
MVS	1.837×10^5	-	-	7.999×10^2

defined as follows:

$$d_{EMD}(X,Y) = \sum_{x \in X} \left[\min_{y \in Y} \{d(x,y)\} \right]. \quad (5)$$

The point at the shortest distance in point cloud Y was calculated from point x, which belonged to point cloud X. The EMD is sum of these distances.

Table I and Table II show the quantitative evaluation on the THU-MVS dataset and the Benchmarking Camera Calibration 2008, respectively. The proposed method was faster than the MVS generation for both datasets. In addition, in the results of the Benchmarking Camera Calibration 2008 in Table II with the same number of vertices, the EMD (avg.) values of uniform distribution were lower than those of the proposed method in the Herz-Jesus-P8 and fountain-P11. Furthermore, the increase in the number of vertices was fewer than that in the THU-MVS dataset.

For the THU-MVS dataset, the model generated by the proposed method was compared with ground truth (GT) provided by the authors. However, the GT model has greater number of vertices than the model generated by the proposed method. Therefore, we simplify the GT mesh model by the quadric edge collapse decimation algorithm of MeshLab [21]. This simplified model is referred as GT (simple). The results are shown in Table III.

TABLE III: **Comparison between the proposed method and ground truth on the THU-MVS dataset.**

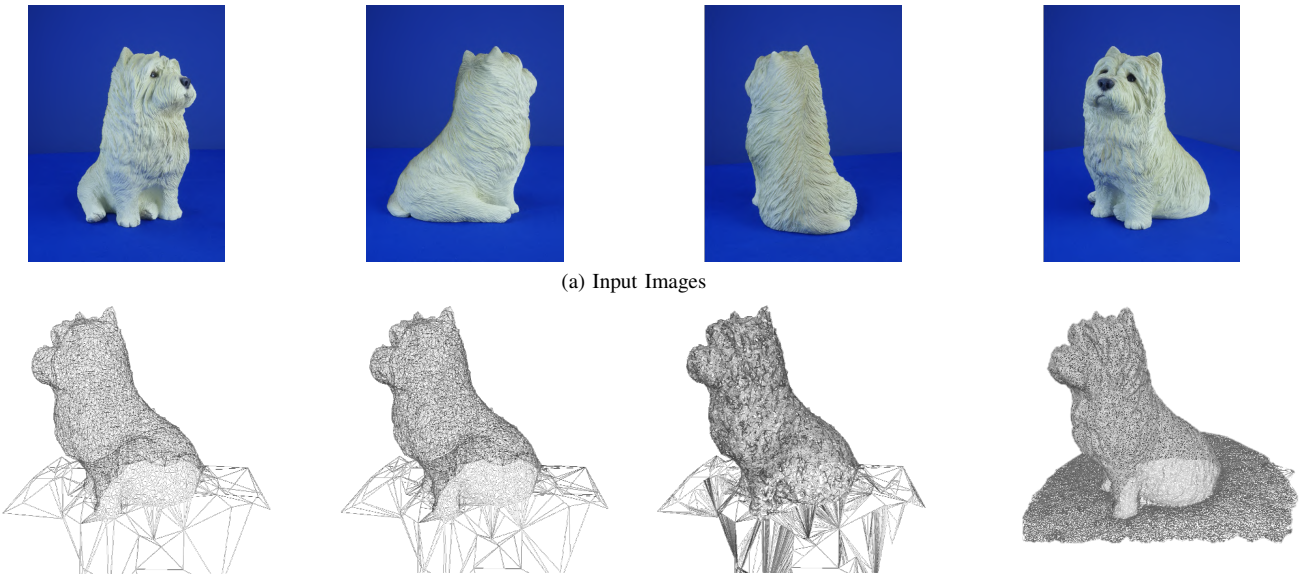
Method	# of Vertices	EMD (avg.)	EMD (var.)
Dog RGB			
GT	8.027×10^5	4.043×10^{-1}	2.910×10^{-2}
GT (simple)	1.223×10^5	3.461×10^{-2}	2.855×10^{-2}

B. Qualitative Evaluation

Qualitative comparisons on the THU-MVS dataset and the Benchmarking Camera Calibration 2008 are shown in Fig. 5 and Fig. 6, respectively. As shown in Fig. 5, the proposed method generates the fine mesh model. In contrast, for complex buildings shown in Fig. 6, because the input models generated through SfM lack some important vertices for representing these shapes, there are some regions where the detailed shapes cannot be obtained by the proposed method.

C. Discussion

As shown in Table II, it was evident that in the Benchmarking Camera Calibration 2008, the EMD (avg.) value of the uniform distribution was lower than that obtained through the proposed method in the Herz-Jesus-P8 and fountain-P11, and the number of vertices was small after the division. This is probably because the data included in the Benchmarking Camera Calibration 2008 were for buildings and several planes were involved. As the estimation result of the normal was correct and even a coarse mosh model correctly represented a plane such as a wall, it was not considered to be the target of division and a division was not necessary. A division was performed in the case of a uniform distribution; however,



(a) Input Images

(b) Coarse Mesh Model (c) Uniform Distribution (d) Proposed Method (e) Ground Truth

Fig. 5: Qualitative evaluation on the THU-MVS dataset.

because division was not performed in the proposed method, the value of the EMD was lower than that of the uniform distribution. Division of a plane implies dividing a part that does not required division. This makes the representation on the mesh redundant and increases the execution time of the mesh's super-resolution. Hence, it can be concluded that the method proposed this study, which divides using normals, is superior to the uniform distribution.

V. Conclusion

In this paper, we proposed a mesh super-resolution method by dividing and deforming a coarse mesh model using the estimated surface normals from images. Experimental results showed that surface normals are effective for mesh deformation.

In the future, we wish to implement the normal estimation from images using neural networks. Furthermore, we aim to improve our method by implementing a recursive triangle mesh division and a deformation algorithm.

Acknowledgment

This work was supported by JSPS KAKENHI Grant Number JP20J13300.

References

[1] D. Maturana and S. Scherer, "VoxNet: A 3D Convolutional Neural Network for real-time object recognition," in *IROS*, pp. 922–928, 2015.

[2] C. R. Qi, H. Su, M. Nießner, A. Dai, M. Yan, and L. J. Guibas, "Volumetric and Multi-view CNNs for Object Classification on 3D Data," *CVPR*, pp. 5648–5656, 2016.

[3] G. Riegler, A. O. Ulusoy, and A. Geiger, "OctNet: Learning Deep 3D Representations at High Resolutions," in *CVPR*, pp. 6620–6629, 2017.

[4] R. Q. Charles, H. Su, M. Kaichun, and L. J. Guibas, "PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation," in *CVPR*, pp. 77–85, 2017.

[5] M. Lhuillier and L. Quan, "A quasi-dense approach to surface reconstruction from uncalibrated images," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 27, no. 3, pp. 418–433, 2005.

[6] L. Yu, X. Li, C. Fu, D. Cohen-Or, and P. Heng, "PU-Net: Point Cloud Upsampling Network," in *CVPR*, pp. 2790–2799, 2018.

[7] W. Yuan, T. Khot, D. Held, C. Mertz, and M. Hebert, "PCN: Point Completion Network," in *3DV*, pp. 728–737, 2018.

[8] E. Kalogerakis, M. Averkiou, S. Maji, and S. Chaudhuri, "3D Shape Segmentation with Projective Convolutional Networks," in *CVPR*, pp. 6630–6639, 2017.

[9] L. Yi, H. Su, X. Guo, and L. Guibas, "SyncSpecCNN: Synchronized Spectral CNN for 3D Shape Segmentation," in *CVPR*, pp. 6584–6592, 2017.

[10] C. Wu, "Towards Linear-Time Incremental Structure from Motion," in *3DV*, pp. 127–134, 2013.

[11] C. Sweeney, T. Sattler, T. Höllerer, M. Turk, and M. Pollefeys, "Optimizing the Viewing Graph for Structure-from-Motion," in *ICCV*, pp. 801–809, 2015.

[12] H. Cui, X. Gao, S. Shen, and Z. Hu, "HSfM: Hybrid Structure-from-Motion," in *CVPR*, pp. 2393–2402, 2017.

[13] F. Langguth, K. Sunkavalli, S. Hadap, and M. Goesele, "Shading-Aware Multi-view Stereo," in *ECCV*, pp. 469–485, 2016.

[14] J. L. Schönberger, E. Zheng, M. Pollefeys, and J.-M. Frahm, "Pixelwise View Selection for Unstructured Multi-View Stereo," in *ECCV*, 2016.

[15] J. L. Schönberger and J.-M. Frahm, "Structure-from-Motion Revisited," in *CVPR*, 2016.

[16] N. Wang, Y. Zhang, Z. Li, Y. Fu, W. Liu, and Y.-G. Jiang, "Pixel2Mesh: Generating 3D Mesh Models from Single RGB Images," in *ECCV*, 2018.

[17] C. Wen, Y. Zhang, Z. Li, and Y. Fu, "Pixel2Mesh++: Multi-View 3D Mesh Generation via Deformation," in *ICCV*, pp. 1042–1051, 2019.

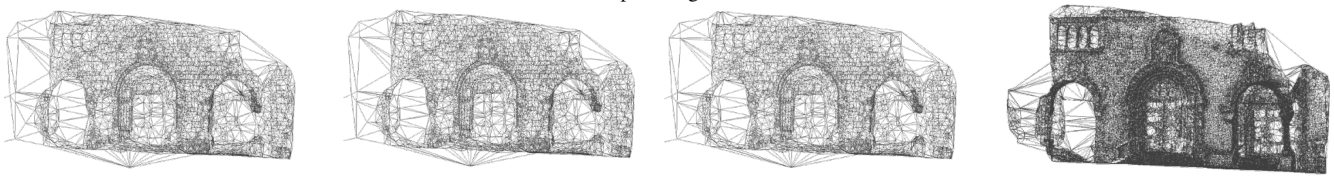
[18] R. Hanocka, A. Hertz, N. Fish, R. Giryes, S. Fleishman, and D. Cohen-Or, "MeshCNN: A Network with an Edge," *ACM Transactions on Graphics (TOG)*, vol. 38, no. 4, p. 90, 2019.

[19] S. Sakai, K. Ito, T. Aoki, T. Watanabe, and H. Unten, "Phase-Based Window Matching with Geometric Correction for Multi-View Stereo," *IEICE Transactions on Information and Systems*, vol. 98, no. 10, pp. 1818–1828, 2015.

[20] C. Strecha, W. von Hansen, L. Van Gool, P. Fua, and U. Thoennessen, "On benchmarking camera calibration and multi-view stereo for high resolution imagery," in *CVPR*, pp. 1–8, 2008.

[21] P. Cignoni, M. Callieri, M. Corsini, M. Dellepiane, F. Ganovelli, and G. Ranzuglia, "MeshLab: an Open-Source Mesh Processing Tool," in *Eurographics Italian Chapter Conference*, pp. 129–136, 2008.

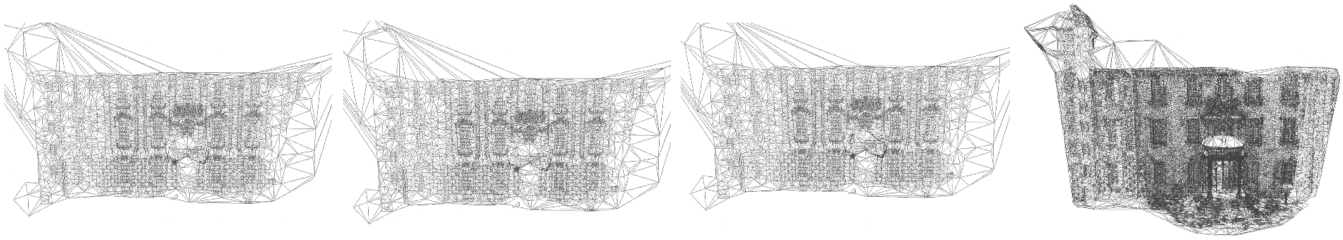
Input Images



Coarse Mesh Model | Uniform Distribution | Proposed Method | MVS

(a) Herz-Jesus-P8

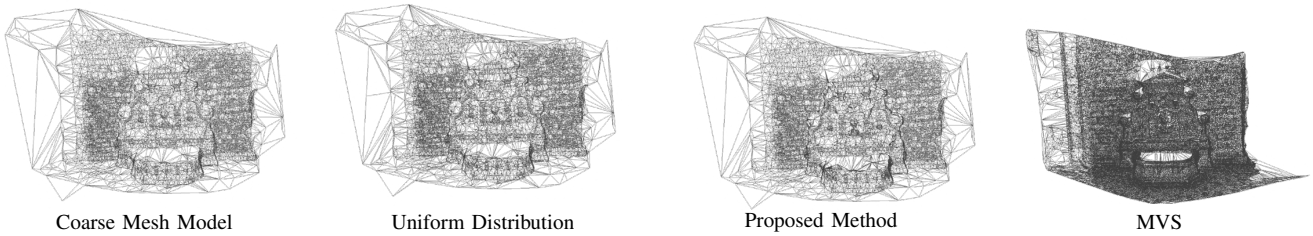
Input Images



Coarse Mesh Model | Uniform Distribution | Proposed Method | MVS

(b) entry-P10

Input Images



Coarse Mesh Model | Uniform Distribution | Proposed Method | MVS

(c) fountain-P11

Fig. 6: Qualitative evaluation on the Benchmarking Camera Calibration 2008.