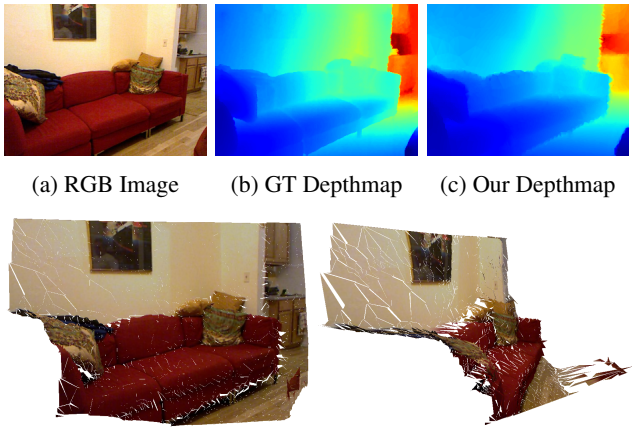
MeshDepth: Disconnected Mesh-based Deep Depth Prediction

Masaya Kaneko[1,2], Ken Sakurada[2], Kiyoharu Aizawa[1]
[1]The University of Tokyo, [2]National Institute of Advanced Industrial Science and Technology (AIST)
[1]{kaneko, aizawa}@hal.t.u-tokyo.ac.jp, [2]k.sakurada@aist.go.jp

Abstract

We propose a novel method for mesh-based single-view depth estimation using Convolutional Neural Networks (CNNs). Conventional CNN-based methods are only suitable for representing simple 3D objects because they estimate the deformation from a predefined simple mesh such as a cube or sphere. As a 3D scene representation, we introduce a disconnected mesh made of 2D mesh adaptively determined on the input image. We made a CNN-based framework to compute depths and normals of faces of the mesh. Because of the representation, our method can handle complex indoor scenes. Using common RGBD datasets, we show that our model achieved best or comparable performance comparing to the state-of-the-art pixel-wise dense methods. It should be noted that our method significantly reduces the number of the parameter representing the 3D structure.



(a) RGB Image (b) GT Depthmap (c) Our Depthmap

(d) 3D Mesh Prediction

Figure 1: We present a novel CNN-based mesh prediction method from a single RGB image. Our results achieve comparable performance to that of the pixel-wise dense method, despite the mesh representation using much less parameters.

1. Introduction

Image-based 3D reconstruction and modeling are important problems for many different applications such as robotics, autonomous vehicles, and augmented reality. Representative techniques are Structure from Motion (SfM) and Multi-View Stereo (MVS), and Simultaneous Localization and Mapping (SLAM).

Many studies have emerged in recent years that use Convolutional Neural Networks (CNNs) for 3D reconstruction. Single-view dense depth map prediction using CNNs is a successful example [3, 4, 19]. A dense depth map obtained from a single view is beneficial to multi-view 3D reconstruction as it can be an initial solution of 3D structure. CNN-based 3D reconstruction generally uses point-cloud representations for the efficient computation. However, there are some drawbacks in point-cloud representations. The number of the parameter is too large and the spatial relationships between the points are not described.

In this paper, we propose a novel CNN-based mesh reconstruction of 3D scenes. In comparison with point-cloud representations a mesh can efficiently represent the 3D structure of an object because it can simplify surfaces (e.g., room wall) and maintain the texture and surface information of the object. To the best of our knowledge, this is the first work in CNN-based 3D reconstruction via a variable mesh representation, whose number of the vertices and faces are adaptive to each scene image.

The problem of mesh estimation itself has been actively studied as surface reconstruction [12, 14, 15]. However, directly applying CNNs to a variable mesh surface estimation is difficult because a mesh is a graph structure composed of nodes and edges, which is incompatible with the general CNNs.

There are existing works on CNN-based mesh estimation, and they are limited to a simple 3DCG model [10, 16, 24]. These studies realize mesh estimation by learning the deformation from a predefined simple mesh model (such as a sphere) using CNNs. It is difficult to use the above method for complex tasks such as general 3D scene reconstruction, because the number of vertices and faces is fixed in advance.

In our proposal, we overcome the problem — we intro-

duce a new representation of a mesh, that is a disconnected mesh, and create a novel CNN architecture to the mesh, whose number of faces and vertices are variable. We realize a CNN-based mesh estimation framework for complex 3D scene reconstruction. We estimate depths and normals of the individual faces of the 2D mesh, as illustrated in Fig. 5. As a result, the 3D mesh is composed of disconnected triangle faces. Each triangle represents a partial surface of the 3D structure. This representation is considered a special case of a mesh, which is effective for the representation of the discontinuity.

The overview of our approach consists of the following elements: (1) A 2D mesh is estimated from an input image based on the intensity-gradient. The mesh size is changeable for each image. (2) The depths and normals of the faces of the 2D mesh are estimated from an RGB image using CNNs. We design a novel CNN architecture that can deal with variable meshes. (3) CNNs are trained using a depth-map-based loss. Since there are almost no 3D mesh datasets for general scenes, we use general RGBD datasets and define the loss between depth map rendered by Neural Mesh Renderer [10] and its ground truth (GT). We evaluate the performance of our method on the NYU Depth v2 [34]. Our method achieves the best or comparable performance to the state-of-the-art pixel-wise dense depth map estimation method. It should be noted that the number of the parameter representing the 3D structure is drastically reduced. The contributions of our work are as follows:

- We introduce a novel representation of 3D structure, disconnected mesh. We parameterize the disconnected mesh by depths and normals of its faces for CNN regression.
- We propose a novel CNN architecture that can deal with the disconnected mesh whose number of faces and vertices are variable. This makes it possible to realize CNN-based mesh estimation for complex 3D shape reconstruction.
- We make an end-to-end training framework which only needs general RGBD datasets without GT mesh data. The performance of our framework is the best or comparable to that of the state-of-the-art pixel-wise-based methods, despite the mesh representation using much less parameters.

2. Related Works

2.1. Pixel-wise Depth Map Prediction

Recently, CNN-based methods have been actively studied for 3D reconstruction. Since CNNs can extract good features from images, the pixel-wise depth map representation is generally used in these works.

Supervised Learning. In recent years, research on depth map prediction from a single-view image with CNNs has been proposed. The supervised learning method with numerous RGBD data is a straight forward approach to single-view depth prediction. The datasets used in the representative works [4, 19] are composed of RGB images and their GT depth maps, which were acquired by Microsoft Kinect for indoors [34] and laser scanners for outdoors [6, 31] . In addition, there is an extended approach to learn the estimation of not only the depth map but also the normal map (or semantic segmentation) simultaneously, which can improve the performance of each task [3, 38]. Under a different problem setting, a work [25] on the densification of a sparse depth map (LiDAR's sparse point cloud or reconstructed 3D map of visual SLAM) is reported. However, the construction of the dataset, which is a key to these supervised methods, is very difficult because of requirements of sensor accuracy, measurement density, and cost. Therefore, with supervised learning it is not realistic to cover all the real world scenarios.

Unsupervised Learning. Therefore, unsupervised learning of a single-view depth prediction has gained attention. In the classical methods in the SfM approach, the 3D structure of an object seen from multiple views is determined based on the 3D geometry [9]. Applying this, unsupervised methods implicitly estimate the depth map and the ego motion from images captured from multiple viewpoints. Two types of images are used: two consecutive images in a time series [44] and left and right images of the stereo camera [7]. Particularly, in a model using the former, the learning is reported to be unstable because its ego motion varies significantly [37]. Various modifications have been proposed for this issue: changing of the CNN architecture [8], applying normalization for the depth map [37], simultaneously predicting the normal map [39, 40], and applying epipolar constraints [27].

Multi-View Integration. As extensions of the above single-view based methods, some works integrate the results of an arbitrary number of multi-view images and estimate 3D structures with CNNs, such as SfM or visual SLAM. Yao *et al.* [41] proposed CNN-based depth map predictor which integrates the features from multi-view images under the assumption that each camera pose is known. As a final goal to realize SfM or visual SLAM, it is required to estimate not only the 3D structure but also the camera trajectory from multi-view images. CNN-SLAM [35] is one solution to this as a robust CNN-based visual SLAM by replacing a part of the existing visual SLAM with a CNN-based depth prediction. Moreover, DeMoN [36] and DeepTAM [43] achieve remarkable performances by realizing the tracking and mapping functions of the visual SLAM with CNNs.

As mentioned above, the CNN-based pixel-wise method

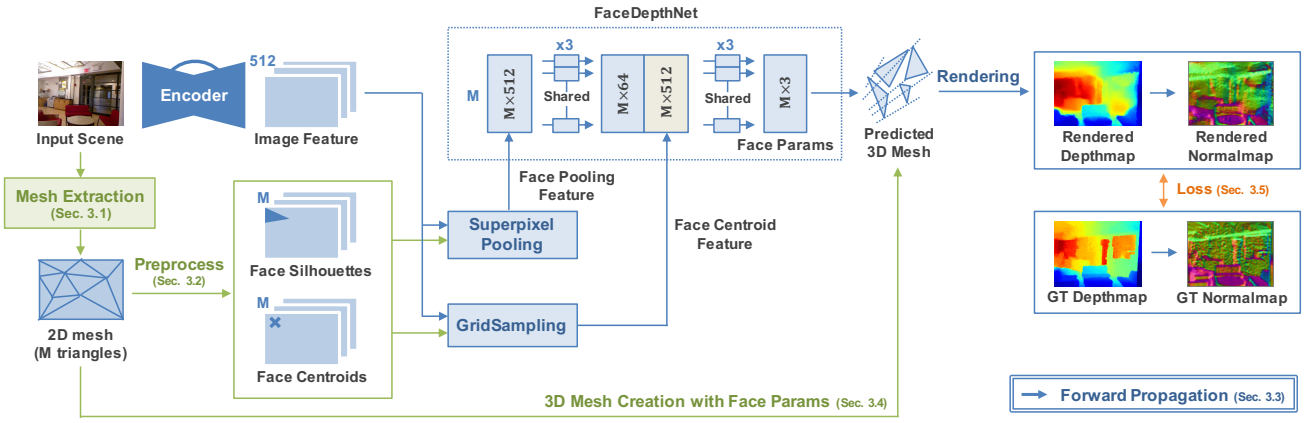


Figure 2: **Illustration of our framework.** After determining a 2D mesh adaptively to the input image (this means that the number of triangles of a 2D mesh, M, varies for each input), CNNs estimate the depths and normals of the faces of the 2D mesh. We train this model end-to-end by back-propagating the loss between the rendered depth map (+ normal map) and its GT.

has made rapid progress. But there are some drawbacks. One of them is that they have too many parameters representing 3D structure for the pixel-wise depth map. The pixel-wise methods are extremely redundant as a representation for planar surfaces. They cannot even retain the texture and surface information of the 3D structure.

2.2. 3D Mesh Estimation

Mesh is one of the representations that can solve the above drawbacks. In fact, a mesh representation is used for 3D modeling in MVS after the point-cloud-based 3D map reconstruction in the SfM. Ideally, the approaches described in the previous section should be performed using a mesh, which would be efficient.

The treatment of meshes with CNNs is a challenging task. Kato *et al.* [10] solved one of the difficulties. They formulated a rasterization of the rendering pipeline in a differentiable form and proposed a novel mesh estimation method based on the rendered image. This work made it easier to optimize a mesh with CNNs. Surface Networks [16] proposed another approach to the mesh estimation using Graph Neural Networks (GNNs), which is a special form of CNNs. Commonly in these methods, mesh estimation is regarded as the deformations of the vertices from a predefined mesh (e.g. cube or sphere) since adaptive changes of the vertices and faces are difficult for CNNs. Their methods are limited to the estimation of a simple 3D mesh such as ShapeNet dataset [2].

In this paper, we propose a novel method that can deal with general 3D reconstruction. Our method can estimate a 3D mesh with different number of faces and vertices for each input image using an end-to-end CNN architecture.

3. Proposed Method

In this section, we describe the overall framework of the proposed method, what we call MeshDepth, for variable mesh estimation, as illustrated in Fig. 2. We explain (1) 2D mesh extraction, (2) its preprocessing for the CNNs, (3) the details of the CNN architecture, (4) the formulation of a disconnected mesh, and (5) the loss function.

3.1. Edge-based Mesh Extraction

The first step of our method is to estimate an appropriate 2D mesh for an input image. In the conventional methods, the number of vertices and faces is fixed by using a predefined mesh model. To cope with the complex shape, we first build an adaptive 2D mesh for an input image following Bódis-Szomorú *et al.* [1]. The edges of the 2D mesh is determined by intensity-gradient of the image. The procedure shown in Fig. 3 is summarized as follows:

1. **Canny Edge Detection.** We apply canny edge detector to the input images.

2. **Edge Simplification.** We simplify the edges by lines and vertices.

3. **Constrained Delaunay Triangulation (CDT).** The Delaunay triangulation is applied to the vertices, maintaining the connected edges.

This operation determines the number of the vertices and faces of a 3D mesh. The two parameters — the input image resolution and the threshold of canny operator — determines the fineness of the 3D mesh. In the default settings, we resize the image to $1/3$ and determine the canny thresholds based on the Otsu method proposed by Fang *et al.* [5].

(a) Scene Image (b) Canny Edge (c) Final 2D mesh

Figure 3: **2D Mesh Extraction.** We construct partially connected vertices based on (b) simplified canny edge from (a) the input image. Applying CDT to it, we can obtain (c) the final 2D mesh.

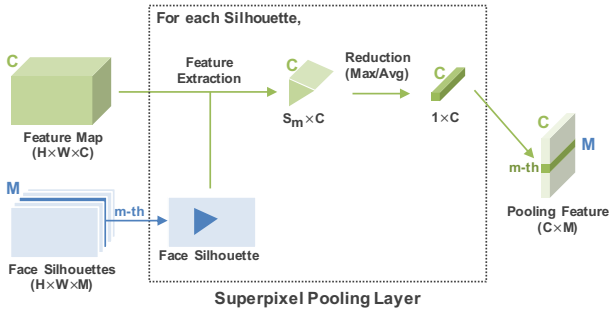


Figure 4: **Overview of the superpixel pooling layer [17].** For each face silhouette (face id is $m \in [1, M]$), feature in the silhouette area ($S_m \times C$) is extracted and reduced for each channel ($1 \times C$). Finally, we can obtain the final feature ($M \times C$) by concatenating them.

3.2. Mesh Preprocessing for CNNs

As described above, our goal is to estimate depths and normals of faces of the 2D mesh. Firstly, we preprocess the 2D mesh for the CNNs (see next section Sec. 3.3 for details) into the following representations:

1. **Face Silhouettes.** A silhouette image, that is a triangle shape, is rendered for each face using the neural mesh renderer [10]. It is used to extract the feature of each face from the encoded image feature.

2. **Face Centroids.** The centroid of each face is calculated. It is used for feature extraction of the centroid position, which does not depend on the face size.

3.3. Network Architecture

We make use of CNNs to estimate the depth and normals of each faces of the mesh. The CNN architecture is described below.

Image Encoder. We first extract deep feature map of the input image. We use DRN-54 [42], which preserves the spatial resolution and increases the resolution of the output feature map. After DRN-54, we construct a feature pyramid network as a decoder to extract rich information from coarse scales to fine scales, following Feature Pyramid Network [21]. By concatenating features from each hierarchy of the pyramid, we finally obtain a 512-dimensional feature map whose resolution is the same as that of the input. The details are illustrated in the supplementary material.

Face Pooling Feature. We extract features corresponding to each face of the 2D mesh. We use Superpixel Pooling Layer [17]. This pooling layer reduces the information from one feature vector per pixel to one feature vector per local region (e.g., superpixel) in the image. In our case, a local region is the face silhouette prepared in Sec. 3.2.

Consider the image feature as $\mathbf{F} \in \mathbb{R}^{C \times P}$ (C channels, $P(= H \times W)$ pixels) and face silhouettes as $\mathbf{S} \in L^P$ ($L \in [1, M]$ denote the face id to which each pixel belongs). As the output of the superpixel pooling layer, $\mathbf{P} \in \mathbb{R}^{C \times M}$ is

$$P_{c,m} = reduce\{F_{c,i} | i : S_i = m\} \tag{1}$$

where *reduce* denotes a function such as *max* or *average* in [17]. In our implementation, we used the *max* function of the simple and efficient GPU-implementation proposed in [32]. The overview is illustrated as Fig. 4.

Face Centroid Feature. In addition to the above feature, we extract the features of the centroid position for each face, which is stable and not affected by the face size. We use the differentiable bilinear sampling technique used in Spatial Transformer Networks [11]. It approximates the feature, F_c (c is the channel id) of the centroid position, $p_m (m \in [1, M]$ is the face id), as the interpolation of four pixel neighbors (top-left, top-right, bottom-left, and bottom-right).

$$P_{c,m} = \sum_{i \in \{t,b\}, j \in \{l,r\}} w^{ij} F_c(p_m^{ij}) \tag{2}$$

where w^{ij} denotes the ratio of the spatial distance to each neighbor ($\sum_{ij} w^{ij} = 1$).

FaceDepthNet. Using features — face pooling feature and face centroid feature —, we finally estimate the parameters representing the 3D scene structure. The face of the disconnected mesh has similar properties to a point of a point cloud because both of them are unordered and interacted each other. Therefore, we created a CNN composed of a shared multi-layer perceptron (MLP) network which is similar to PointNet [28]. The architecture difference between ours and [28] is that we exclude the max pooling and feature transform, since the global context information has already been obtained by the image encoder. The detailed network architecture is illustrated in the supplementary material.

3.4. Formulation of Disconnected Mesh

In this section, we explain how to determine the 3D mesh structure from the parameters estimated by FaceDepthNet. For each face (face id $m \in [1, M]$), FaceDepthNet outputs the following three parameters (d_m, θ_m, ϕ_m):

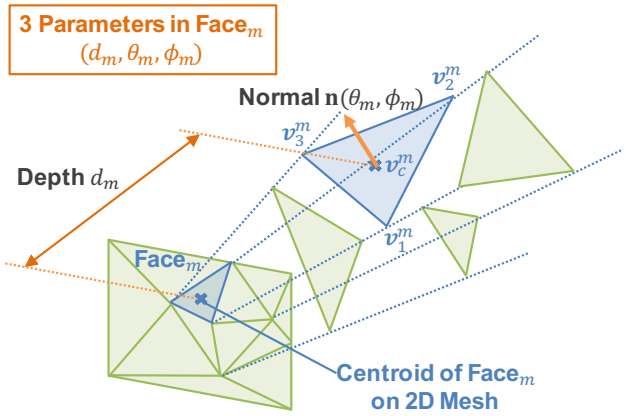


Figure 5: **Disconnected Mesh.** For each face, three parameters are used to determine its 3D position. The 3D face triangles are treated independently for the effective representation of complex 3D shapes.

1. Face depth, d_m, at the centroid position (u_m, v_m) in the image coordinates.
2. Face normal angle, (θ_m, ϕ_m), in spherical coordinates, whose origin is $[u_m, v_m, d_m]^{\mathrm{T}}$.

These definitions are illustrated in Fig. 5. With these parameters, we can uniquely determine the planar equation of a face in 3D space. Each depth value per pixel, (x, y), in the face silhouette region, S_m, can be determined by the following formula:

$$D_m(x,y) = \begin{bmatrix} -\tan\theta_m \cos\phi_m \\ -\tan\theta_m \sin\phi_m \\ d_m \end{bmatrix}^{\mathrm{T}} \begin{bmatrix} x - u_m \\ y - v_m \\ 1 \end{bmatrix} \quad (3)$$

where $\theta_m \in [-\pi/2, \pi/2]$, $\phi_m \in [0, 2\pi)$ and $(x, y) \in S_m$. Once the depth of a vertex, whose image coordinates are $\mathbf{u} = (x, y)$, is determined, we can transform it into the 3D coordinates, $V(\mathbf{u})$, with a camera intrinsic matrix, $\mathbf{K}$.

$$V(\mathbf{u}) = \mathbf{K}^{-1}\dot{\mathbf{u}} D_m(\mathbf{u}) \quad (4)$$

where $\dot{\mathbf{u}}$ denotes its homogeneous representation. We can determine the 3D structure of the mesh from the vertices in the 3D coordinates.

Note that the 3D mesh obtained by our formulation, what we call a disconnected mesh, ignores the adjacency connection between the faces, and the faces are independent triangles. Although this formulation loses smoothness and increases the number of parameters compared to a general mesh, it has the following advantages: (1) It is compatible to a CNN regression like our proposal, and (2) it is effective representation for complex shapes, such as occlusion, which is commonly found in a 3D scene structure. Furthermore, considering the application to visual SLAM, the disconnected meshes of multiple viewpoints are easy to register and integrate into a 3D map.

3.5. Loss Function

In this section, we explain CNN optimization of the 3D mesh. For versatility, we aim to optimize our framework using a general RGBD dataset. We define the loss from the mean absolute error (L_1) between the GT depth map and rendered depth map from the 3D mesh by [10], which is similar to the pixel-wise single-view depth prediction method [3, 4, 19].

In addition, since the normal direction is used in the mesh parameterization, we include normal in our loss function. We use a method proposed in [40] to construct a differentiable normal map by computing the normal of each pixel from eight pixel neighbors in the depth map.

Summarizing the above, we define the loss function from the rendered depth map D and GT depth map D^*, and the corresponding normal map N and GT normal map N^*, as follows.

$$\begin{aligned} L_{sum} &= L_{depth} + \lambda_n L_{normal} \\ &= \frac{1}{n}\sum_i (D_i - D_i^*) + \lambda_n (-\frac{1}{n}\sum_i (N_i \cdot N_i^*)) \end{aligned} \quad (5)$$

where i is the valid pixel id, n is the total number of valid depth pixels, and λ_n is a balancing factor, which is tuned with a sampled validation set from the training images (we use $\lambda_n = 0.5$ as the best value).

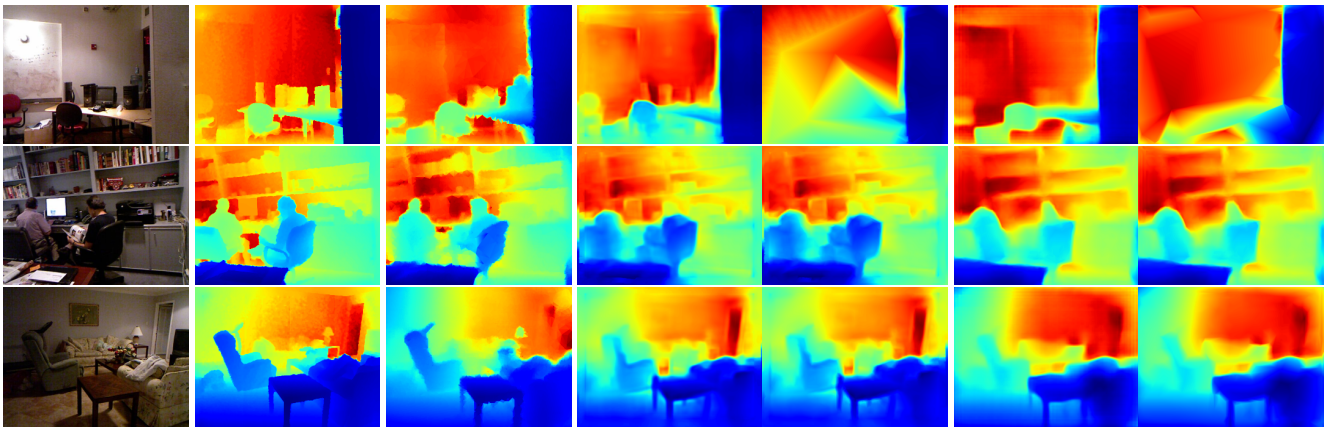
4. Experimental Results

In this section, we present an analysis of the results of the proposed method. In Sec. 4.1 and Sec. 4.2, we explain the detail of our implementation and experimental settings for the training. In Sec. 4.3, we report the quantitative and qualitative results obtained by our method and compare them with those of the existing state-of-the-art methods. We also compare our results with the naive mesh-based method.

In addition, we present a more detailed analysis of the proposed method. First, we describe ablation study to analyze the influence of each factor of the proposed method (Sec. 4.4). We also discuss the investigation of the robustness of our CNNs to the fineness of the 2D mesh (Sec. 4.5).

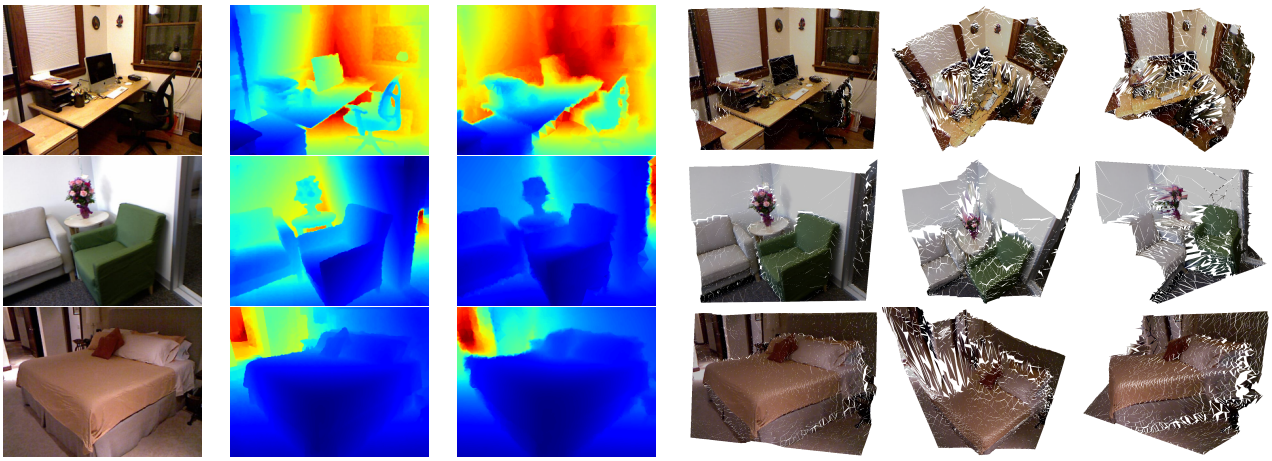
4.1. Experimental Set-up

We used the PyTorch framework [26] for the implementation of our CNN model. We implemented an image encoder based on DRN-54 [42] initialized with the pre-trained weights on ImageNet [30], and applied a face pooling feature extractor using the GPU-implemented superpixel pooling [32], as discussed in Sec. 3.3. In other parts, for exam-



(a) Input (b) GT (c) **Ours** (d) Eigen *et al.* [3] (original, naive) (e) Laina *et al.* [19] (original, naive)

Figure 6: **Depth Map Prediction.** Qualitative results showing our depth map results, the results of the pixel-wise-based methods [3, 19] (left) and their mesh-based naive methods (right). Each depth map is scaled for better visualization.



(a) Input (b) GT Depthmap (c) Our Depthmap (d) 3D Mesh Prediction

Figure 7: **3D Mesh Prediction.** (a) is an RGB input image and (d) are the multi-views of the 3D mesh predicted from (a). (b) is a GT depth map of the input scene and (c) is a rendered depth map from (d).

ple the 2D mesh extractor (Sec. 3.1), we used OpenCV for the canny edge extraction and Triangle [33] for CDT. We used the neural mesh renderer [10] for the mesh rendering parts, as described in Sec. 3.2 and Sec. 3.5. We trained the above implemented model on a single NVIDIA Tesla P40 with 24GB of GPU memory.

This network was trained to predict the depth map using RGB images as the input, similar to single-view depth prediction networks [3, 4, 19]. To increase the number of training samples, we augmented them with some random transformations. We used small rotations, scaling, color jitter, color normalization and flips with 0.5 chance, following the settings of [25]. As qualitative evaluation metrics, we used the same metrics that have been used in previous works [3, 4, 18, 19, 20, 22] for performance comparison.

4.2. Dataset for Evaluation

We trained our MeshDepth using the NYU depth v2 [34], which is one of the largest RGBD datasets for indoor scene reconstruction. This indoor dataset satisfies the requirement that GT depth map should be maximally dense to calculate the normal map for loss function (Eq. (5)).

This dataset is composed of pairs of an RGB image and the depth image of 464 scenes captured by Microsoft Kinect. We followed the official splitting, 249 scenes for training, and 215 scenes for testing. For this evaluation, we used approximately 48K pairs (RGB images and their corresponding GT depth maps), which are sampled spatially uniformly from the scenes in the raw training dataset for training, and used 654 labelled images for evaluating

the final performance. We used the preprocessed dataset distributed by Ma *et al.* [25]. They created synchronized RGBD image pairs, projecting each depth value onto the RGB image and in-painting them with the official toolbox. Following previous works [3, 19, 25], the image input in the network was down-sampled to half from the original resolution (480×640) and center-cropped to 228×304. We trained our MeshDepth for approximately 50 epochs with a batch-size of 4. We used the Stochastic Gradient Descent (SGD) optimizer with learning rate of 10^{-3}, momentum of 0.9, and weight-decay of 10^{-4}. In addition, we used 1% of the training dataset separately as a validation dataset for the hyperparameter search. The final score of the test dataset was evaluated using the trained model with the highest validation score.

4.3. Comparison with Existing Methods

Pixel-wise-based methods. First, we compare our proposed method with the existing pixel-wise-based methods. As explained in the previous section, we calculate the score with the general error metrics (RMSE, REL, $\log_{10}$, $\delta_1 \sim \delta_3$) used in pixel-wise-based single-view depth prediction [3, 4, 18, 19, 20, 22]. To perform evaluation similar to the pixel-wise-based methods, we use the depth map rendered from the estimated 3D mesh (see Fig. 7). The results are provided in Table 1.

Our MeshDepth achieved the best or comparable performance to the state-of-the-art pixel-wise-based methods. It should be noted that the number of the parameter representing the 3D structure was drastically reduced. The parameter size implies the size of the map points registered in the 3D map in the form of a point cloud. Since this evaluation is performed with the original resolution (480×640), the parameter size of the pixel-wise-based methods is 921K($= 480 \times 640 \times 3$). Comparatively, in our method, three vertices are held for each face, so that the number of the parameter is (the average number of faces) $\times 3$ vertices $\times$ 3, and, in this case, it is about 32K($= 3535.34 \times 9$).

Naive mesh-based methods. There is no mesh-based 3D scene reconstruction using CNNs. Then, we compare the performance of our method with those of the naive mesh-based methods. Here, a naive method involves the following procedure: (1) A dense mesh is constructed by connecting the adjacent vertices of the dense depth map obtained by a pixel-wise based method. (2) The dense mesh is simplified until it has the same number of faces as our method. In this evaluation, we used the two state-of-the-art works (Engel *et al.*[3] and Laina *et al.*[19]) as the pixel-wise-based methods to construct a dense depth map. Both works distribute the prediction results for the test set of the NYU depth v2. Our evaluation follows the procedure described in the previous subsection by rendering the depth map from the estimated 3D mesh. For the invalid areas (outside of the rendered silhouette), we inpaint them using OpenCV and create dense depth maps.

Table 1: **Comparison with pixel-wise-based methods.** Having low error metrics (REL, RMSE, $\log_{10}$) and high accuracy metrics ($\delta_1 \sim \delta_3$) is advantageous. These scores are reported by the authors in their respective papers.

Method	rel	rms	$\log_{10}$	δ_1	δ_2	δ_3	#param.
Karsch *et al.* [13]	0.374	1.12	0.134	-	-	-	921K
Ladicky *et al.* [18]	-	-	-	0.542	0.829	0.941	921K
Liu *et al.* [23]	0.335	1.06	0.127	-	-	-	921K
Li *et al.* [20]	0.232	0.821	0.094	0.621	0.886	0.968	921K
Liu *et al.* [22]	0.230	0.824	0.095	0.614	0.883	0.971	921K
Wang *et al.* [38]	0.220	0.745	0.094	0.605	0.890	0.970	921K
Eigen *et al.* [4]	0.215	0.907	-	0.611	0.887	0.971	921K
Roy and Todorovic [29]	0.187	0.744	0.078	-	-	-	921K
Eigen and Fergus [3]	0.158	0.641	-	0.769	0.950	**0.988**	921K
Laina [19]	**0.127**	0.573	**0.055**	**0.811**	0.953	**0.988**	921K
Ours	0.146	**0.530**	0.062	0.803	**0.954**	**0.988**	**32K**

Table 2: **Comparison with naive mesh-based methods.** These naive methods use the predictions provided by the authors.

Method	rel	rms	$\log_{10}$	δ_1	δ_2	δ_3
ver. Eigen and Fergus [3]	0.163	0.559	0.069	0.762	0.948	0.987
ver. Laina [19]	0.154	0.535	0.064	0.793	0.949	0.987
Ours	**0.146**	**0.530**	**0.062**	**0.803**	**0.954**	**0.988**

Table 3: **Ablation study.** Addition of more factors enhances the performance.

Method	rel	rms	$\log_{10}$	δ_1	δ_2	δ_3
Ours (w/o Centroid, Normal)	0.165	0.559	0.068	0.771	0.946	0.986
Ours (w/o Centroid)	0.155	0.546	0.065	0.782	0.953	**0.988**
Ours	**0.146**	**0.530**	**0.062**	**0.803**	**0.954**	**0.988**

The results are shown in Table 2. They show that our method was able to achieve higher accuracy than both for all the evaluation metrics. Furthermore, according to the qualitative results displayed in Fig. 6, the naive methods often failed in optimization in the simplification process (see the top line of Fig. 6), and our method can estimate the depth map that reflects the object boundaries clearly. This result demonstrates that our disconnected mesh representation formulation is effective for complex 3D scenes.

4.4. Ablation Study

Our method, MeshDepth, is composed of various factors introduced in Sec. 3. We conduct experiments in an ablation study to determine the contributions of these factors to the performance. We analyze the following two elements:

1. Face centroid feature, as discussed in Sec. 3.3, which is used as a stable feature independent of the face size.

2. Normal loss, L_{normal}, as expressed in Eq. (5) in Sec. 3.5 for the optimization on the normal map.

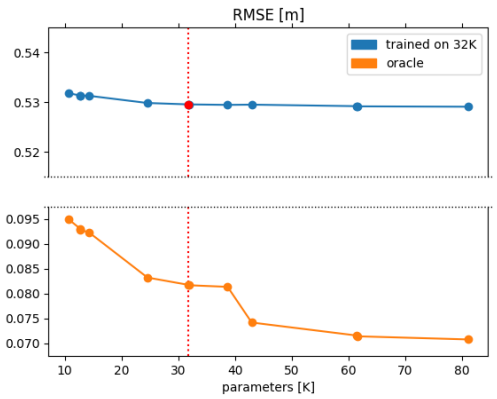


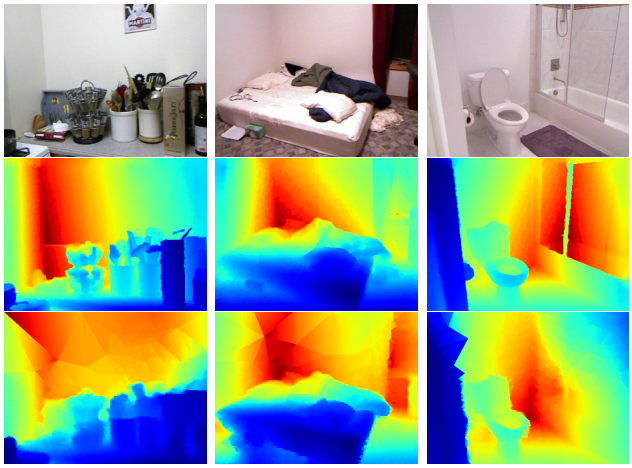
Figure 8: Robustness to the parameter change of the 2D mesh extraction. The performance does not deteriorate even in the fineness which is not used for training, and the performance slightly improves as the number of parameters increases.

We train our model without the factors and investigate the performance of each trained model in the NYU depth v2. The results are listed in Table 3. Based on the results, the performance improved as more factors were added.

4.5. Robustness to the 2D Mesh Extraction

In our framework, the 2D mesh extraction, which determines the size of the final 3D mesh (fineness), is one of the most important factors. As mentioned in Sec. 3.1, we train our MeshDepth with a fixed default setting. The model is required not to deteriorate the performance as much as possible even if the fineness of 2D mesh extraction is changed.

We examine the effect of the fineness change of the 2D mesh extraction on the performance of the model trained by the fixed fineness. Fig. 8 shows the result of this experiment using the evaluation metric, RMSE. First, the orange line represents the limit performance (oracle). According to each fineness, the oracle denotes the accuracy of the depth map obtained by optimizing the parameters in each fineness to minimize the error associated with the GT depth map. Therefore, a high fineness (large number of parameters) leads to an improved oracle value. On the other hand, the blue line represents the change in the inference performance of our model at each fineness. In the default settings that we used for the training, the number of parameters was approximately 32K, which is indicated by a red dot on the blue line. The results show that our method does not deteriorate as the number of parameters changes, but it slightly improved. This suggests that our method can perform robust against the changes in the fineness of the 2D mesh extraction.



(a) Case 1 (b) Case 2 (c) Case 3

Figure 9: **Limitation of Our Method.** These are examples of scenes where accurate 2D mesh extraction cannot be performed with a canny edge, and so 3D mesh prediction cannot be conducted correctly.

5. Conclusion

In this work, we present a novel approach to the problem of 3D reconstruction for general scene. Unlike the existing methods, our method can estimate the 3D mesh, whose number of faces and vertices changes according to the input image. We introduce a disconnected mesh made of 2D mesh adaptively determined on the input image. And we made a CNN-based framework to compute depths and normals of faces of the mesh. This framework, MeshDepth, can be trained end-to-end from pairs of scene images and their GT depth maps in general RGBD datasets. The predictions achieved the best or comparable performance to those of the state-of-the-art pixel-wise dense methods. It should be noted that the size of the parameter representing the 3D structure was drastically reduced.

There is a limitation in our method. In this work, the 2D mesh extraction was conducted based on a canny edge, and there are some cases where this extraction does not work well. For example, Fig. 9 shows failure scenes where the brightness change is not very clear, and the final prediction is inaccurate because the 2D mesh is not appropriately configured. This phenomenon is noticeable outdoors, where the lighting change is large. In the future work, we plan to make 2D mesh extraction more robust by incorporating it into the CNN architecture for end-to-end training.

References

[1] A. Bdis-Szomor, H. Riemenschneider, and L. Van Gool. Efficient Edge-aware Surface Mesh Reconstruction for Urban Scenes. *CVIU*, 157(C), 2017. 3

[2] A. X. Chang, T. A. Funkhouser, L. J. Guibas, P. Hanrahan, Q.-X. Huang, Z. Li, S. Savarese, M. Savva, S. Song, H. Su, J. Xiao, L. Yi, and F. Yu. ShapeNet: An Information-Rich 3D Model Repository. *arXiv:1512.03012*, 2015. 3

[3] D. Eigen and R. Fergus. Predicting Depth, Surface Normals and Semantic Labels with a Common Multi-scale Convolutional Architecture. In *ICCV*, 2015. 1, 2, 5, 6, 7

[4] D. Eigen, C. Puhrsch, and R. Fergus. Depth Map Prediction from a Single Image using a Multi-Scale Deep Network. In *NIPS*, 2014. 1, 2, 5, 6, 7

[5] M. Fang, G. Yue, and Q. Yu. The Study on An Application of Otsu Method in Canny Operator. 01 2009. 3

[6] A. Geiger, P. Lenz, and R. Urtasun. Are we ready for Autonomous Driving? The KITTI Vision Benchmark Suite. In *CVPR*, 2012. 2

[7] C. Godard, O. Mac Aodha, and G. J. Brostow. Unsupervised Monocular Depth Estimation with Left-Right Consistency. In *CVPR*, 2017. 2

[8] C. Godard, O. Mac Aodha, and G. J. Brostow. Digging Into Self-Supervised Monocular Depth Estimation. *arXiv:1806.01260*, 2018. 2

[9] R. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. 2 edition, 2003. 2

[10] H. Kato, Y. Ushiku, and T. Harada. Neural 3D Mesh Renderer. In *CVPR*, 2018. 1, 2, 3, 4, 5, 6

[11] M. Jaderberg, K. Simonyan, A. Zisserman, and K. Kavukcuoglu. Spatial Transformer Networks. In *NIPS*, 2015. 4

[12] M. Jancosek and T. Pajdla. Exploiting Visibility Information in Surface Reconstruction to Preserve Weakly Supported Surfaces. *ISRN*, 2014:1–20, 2014. 1

[13] K. Karsch, C. Liu, and S. B. Kang. Depth Extraction from Video Using Non-parametric Sampling. In *ECCV*, 2012. 7

[14] M. Kazhdan, M. Bolitho, and H. Hoppe. Poisson Surface Reconstruction. In *SGP*, 2006. 1

[15] M. Kazhdan and H. Hoppe. Screened Poisson Surface Reconstruction. *ACM TOG*, 32(3), 2013. 1

[16] I. Kostrikov, Z. Jiang, D. Panozzo, D. Zorin, and B. Joan. Surface Networks. In *CVPR*, 2018. 1, 3

[17] S. Kwak, S. Hong, and B. Han. Weakly Supervised Semantic Segmentation Using Superpixel Pooling Network. In *AAAI*, 2017. 4

[18] L. Ladicky, J. Shi, and M. Pollefeys. Pulling Things out of Perspective. In *CVPR*, 2014. 6, 7

[19] I. Laina, C. Rupprecht, V. Belagiannis, F. Tombari, and N. Navab. Deeper Depth Prediction with Fully Convolutional Residual Networks. In *3DV*, 2016. 1, 2, 5, 6, 7

[20] B. Li, C. Shen, Y. Dai, A. Hengel, and M. He. Depth and surface normal estimation from monocular images using regression on deep features and hierarchical CRFs. In *CVPR*, 2015. 6, 7

[21] T. Lin, P. Dollár, R. B. Girshick, K. He, B. Hariharan, and S. J. Belongie. Feature Pyramid Networks for Object Detection. In *CVPR*, 2017. 4

[22] F. Liu, C. Shen, and G. Lin. Deep Convolutional Neural Fields for Depth Estimation from a Single Image. In *CVPR*, 2015. 6, 7

[23] M. Liu, M. Salzmann, and X. He. Discrete-Continuous Depth Estimation from a Single Image. In *CVPR*, 2014. 7

[24] S. Liu, W. Chen, T. Li, and H. Li. Soft Rasterizer: Differentiable Rendering for Unsupervised Single-View Mesh Reconstruction. *arXiv:1901.05567*, 2019. 1

[25] F. Ma and S. Karaman. Sparse-to-Dense: Depth Prediction from Sparse Depth Samples and a Single Image. In *ICRA*, 2018. 2, 6, 7

[26] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer. Automatic differentiation in PyTorch. In *NIPSW*, 2017. 5

[27] V. Prasad and B. Bhowmick. SfMLearner++: Learning Monocular Depth & Ego-Motion Using Meaningful Geometric Constraints. In *WACV*, 2019. 2

[28] C. R. Qi, H. Su, K. Mo, and L. J. Guibas. PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation. In *CVPR*, 2017. 4

[29] A. Roy and S. Todorovic. Monocular Depth Estimation Using Neural Regression Forest. In *CVPR*, 2016. 7

[30] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei. ImageNet Large Scale Visual Recognition Challenge, 2014. 5

[31] A. Saxena, M. Sun, and A. Y. Ng. Make3D: Learning 3D Scene Structure from a Single Still Image. *TPAMI*, 31(5), 2009. 2

[32] M. Schuurmans, M. Berman, and M. B. Blaschko. Efficient semantic image segmentation with superpixel pooling. *arXiv:1806.02705*, 2018. 4, 5

[33] J. R. Shewchuk. Triangle: Engineering a 2D Quality Mesh Generator and Delaunay Triangulator. In *WACG*, volume 1148, 1996. 6

[34] N. Silberman, D. Hoiem, P. Kohli, and R. Fergus. Indoor Segmentation and Support Inference from RGBD Images. In *ECCV*, 2012. 2, 6

[35] K. Tateno, F. Tombari, I. Laina, and N. Navab. CNN-SLAM: Real-Time Dense Monocular SLAM with Learned Depth Prediction. In *CVPR*, 2017. 2

[36] B. Ummenhofer, H. Zhou, J. Uhrig, N. Mayer, E. Ilg, A. Dosovitskiy, and T. Brox. DeMoN: Depth and Motion Network for Learning Monocular Stereo. In *CVPR*, 2017. 2

[37] C. Wang, J. Miguel Buenaposada, R. Zhu, and S. Lucey. Learning Depth From Monocular Videos Using Direct Methods. In *CVPR*, 2018. 2

[38] P. Wang, X. Shen, Z. Lin, S. Cohen, B. Price, and A. Yuille. Towards Unified Depth and Semantic Prediction from a Single Image. In *CVPR*, 2015. 2, 7

[39] Z. Yang, P. Wang, Y. Wang, W. Xu, and R. Nevatia. LEGO: Learning Edge with Geometry all at Once by Watching Videos. In *CVPR*, 2018. 2

[40] Z. Yang, P. Wang, W. Xu, L. Zhao, and R. Nevatia. Unsupervised Learning of Geometry From Videos With Edge-Aware Depth-Normal Consistency. In *AAAI*, 2018. 2, 5

[41] Y. Yao, Z. Luo, S. Li, T. Fang, and L. Quan. MVSNet: Depth Inference for Unstructured Multi-view Stereo. In *ECCV*, 2018. 2

[42] F. Yu, V. Koltun, and T. Funkhouser. Dilated Residual Networks. In *CVPR*, 2017. 4, 5

[43] H. Zhou, B. Ummenhofer, and T. Brox. DeepTAM: Deep Tracking and Mapping. In *ECCV*, 2018. 2

[44] T. Zhou, M. Brown, N. Snavely, and D. G. Lowe. Unsupervised Learning of Depth and Ego-Motion from Video. In *CVPR*, 2017. 2