IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received September 4, 2019, accepted October 9, 2019, date of publication October 14, 2019, date of current version October 28, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2947195

Octave Deep Plane-Sweeping Network: Reducing Spatial Redundancy for Learning-Based Plane-Sweeping Stereo

REN KOMATSU[1], (Student Member, IEEE), HIROMITSU FUJII[2], (Member, IEEE), YUSUKE TAMURA[1], (Member, IEEE), ATSUSHI YAMASHITA[1], (Member, IEEE), AND HAJIME ASAMA[1], (Fellow, IEEE)

[1]Graduate School of Engineering, The University of Tokyo, Tokyo 113-8656, Japan
[2]Faculty of Advanced Engineering, Chiba Institute of Technology, Narashino 275-0016, Japan

Corresponding author: Ren Komatsu (komatsu@robot.t.u-tokyo.ac.jp)

ABSTRACT In this paper, we propose the octave deep plane-sweeping network (OctDPSNet). OctDPSNet is a novel learning-based plane-sweeping stereo, which drastically reduces the required GPU memory and computation time while achieving a state-of-the-art depth estimation accuracy. Inspired by octave convolution, we divide image features into high and low spatial frequency features, and two cost volumes are generated from these using our proposed plane-sweeping module. To reduce spatial redundancy, the resolution of the cost volume from the low spatial frequency features is set to half that of the high spatial frequency features, which enables the memory consumption and computational cost to be reduced. After refinement, the two cost volumes are integrated into a final cost volume through our proposed pixel-wise ''squeeze-and-excitation'' based attention mechanism, and the depth maps are estimated from the final cost volume. We evaluate the proposed model on five datasets: SUN3D, RGB-D SLAM, MVS, Scenes11, and ETH3D. Our model outperforms previous methods on five datasets while drastically reducing the memory consumption and computational cost. Our source code is available at `https://github.com/matsuren/octDPSNet`.

INDEX TERMS Convolutional neural networks, deep neural networks, depth reconstruction, plane-sweeping stereo, stereo vision.

I. INTRODUCTION

Depth estimation is a fundamental task in the fields of computer vision and robotics, especially for autonomous navigation or autonomous driving, as it is necessary to understand the surrounding environments. RGB cameras, RGB-D cameras, and LiDAR are commonly employed for depth estimation. Among these, RGB cameras are the most popular sensors owing to their low cost, light weight, and availability.

Depth estimation from multi-view images has been comprehensively studied over a long period [1]–[5]. One method is plane-sweeping stereo, where multi-view images are projected onto virtual planes at several distances from the reference image plane to generate a cost volume. Then, the depth maps are estimated using this cost volume.

Recently, several learning-based plane-sweeping stereo methods have been proposed and have achieved high accuracies, owing to the advances in deep learning [6]–[12]. Because convolutional neural networks (ConvNets) can exploit the context information in whole images, they have successfully reconstructed depth maps, even in regions with poor texture, where it is difficult for conventional methods to estimate depth maps accurately.

In state-of-the-art learning-based plane-sweeping stereo methods [6]–[9], the cost volume is represented by a four-dimensional (4D) volume (feature width $\times$ feature height $\times$ number of virtual planes $\times$ feature channel), and so three-dimensional (3D) ConvNets are applied for cost volume regularization. Although 3D ConvNets are beneficial for enhancing the depth estimation accuracy, they require significant memory and computational costs, which is undesirable in some cases. For example, MVSNet [8] requires several seconds for depth estimation, and thus it is difficult to utilize in robotics (e.g., autonomous navigation). Although

The associate editor coordinating the review of this manuscript and approving it for publication was Qichun Zhang.

This work is licensed under a Creative Commons Attribution 4.0 License. For more information, see http://creativecommons.org/licenses/by/4.0/

DPSNet [9] requires less time (0.5 s) for depth estimation, it requires a significant amount of memory during training. Therefore, a faster inference speed and lower memory consumption are necessary for learning-based plane-sweeping stereo while still achieving accurate results.

Although one study [13] has demonstrated that the more parameters a network has the more accurate results it achieves, several studies have been conducted on improving the accuracy while increasing the inference speed and decreasing the model size. Octave convolution (OctConv) [14] is one method that focuses on reducing the spatial redundancy. Here, features are split into high and low frequencies. The high frequencies contain fine information, whereas the low frequencies contain rough information. To reduce the spatial redundancy, a smaller spatial resolution was utilized for the low frequencies, and both the memory consumption and computational cost were successfully reduced without degrading the accuracy.

In this paper, a novel learning-based plane-sweeping stereo is proposed, which we call the octave deep plane-sweeping network (OctDPSNet). Our motivation is that we reduce the resolution of the cost volume to deal with the memory consumption and computational cost problems. However, just reducing the resolution may cause the performance degradation. Therefore, inspired by OctConv, we focus on the spatial redundancy of the cost volume. We divide image features into high and low spatial frequency features, where the resolution of the latter is smaller. Furthermore, two cost volumes (high-frequency and low-frequency cost volumes) are generated by the respective features in our proposed plane-sweeping method. In the plane-sweeping, the number of virtual planes for the low-frequency cost volume is set to half that for the high-frequency cost volume, to reduce the spatial redundancy. Consequently, the memory consumption and computational cost for cost volume regularization are significantly decreased. After cost volume regularization, the two cost volumes are integrated into a single final cost volume. Although addition could be used to integrate the two cost volumes, this may result in undesirable effects, as the high-frequency cost volume contains fine structures (e.g., edges or small depth changes) of scenes in addition to some noise, which might degrade the accuracy of the estimated depth. On the other hand, the low-frequency cost volume contains the rough structure. Therefore, a new integration module is proposed to consider the importance of the two cost volumes at a certain location. A pixel-wise "squeeze-and-excitation" (SE)-based [15] attention mechanism is utilized in the integration module. Finally, the depth maps are estimated from the integrated cost volume using a soft argmax module.

Therefore, our main contribution is to divide the cost volume into two parts: a high-frequency part that includes fine-structure information and a low-frequency part that includes rough-structure information. A smaller resolution is set for the low-frequency cost volume to reduce the spatial redundancy, which enables the memory consumption and computational cost to be reduced. To accomplish this, a new plane-sweeping module and new integration module are proposed.

The remainder of this paper is organized as follows. Related work is briefly reviewed in Section II. Then, we introduce our model in Section III. We describe the experiments for evaluating our method in Section IV, and the results are presented in Section V. Finally, we discuss our results in Section VI and conclude the paper in Section VII.

II. RELATED WORK

In learning-based plane-sweeping stereo methods, some methods [6], [7] only accept rectified stereo-pair images. GC-Net [6] first applied 3D ConvNets to cost volume regularization, and a differentiable soft argmin module was introduced to estimate continuous depth maps from the cost volume. PSMNet [7] introduced a spatial pyramid pooling (SPP) module and stacked hourglass 3D ConvNet module to exploit the context information in the whole image.

Other learning-based plane-sweeping stereo methods [8]–[12] accept unstructured multiple images with relative camera poses. These are intended to be combined with structure-from-motion (SfM) or visual simultaneous localization and mapping (vSLAM) to obtain the relative camera poses.

DeepMVS [10] introduced image patch matching networks to estimate depth maps, which enables an arbitrary number of images to be taken as input. Because image patches (e.g., size 128 $\times$ 128) are utilized to estimate patches of depth maps, it is difficult to exploit the context information in the whole image. Therefore, the authors applied conditional random fields (CRFs) [16] to refine the noisy depth maps. Furthermore, the computational speed is slow owing to the matching-based approach and postprocessing using CRFs.

MVSNet [8] introduced a differentiable homography warping for end-to-end training and achieved a state-of-the-art depth estimation accuracy. However, this method consumes a significant amount of memory and requires several seconds for depth estimation.

To address the memory problem, the authors of MVSNet proposed R-MVSNet [11], which applies a gate recurrent unit (GRU) instead of 3D ConvNet for the cost volume regularization. While the authors reduces the memory consumption, the accuracy is lower than that using 3D ConvNets. In addition, the authors did not take the inference speed into account, and therefore several seconds were required for the depth estimation.

MVDepthNet [12] focuses on the inference speed, so that it can be combined with vSLAM. The authors represented the cost volume as a 3D rather than 4D volume to avoid using 3D ConvNets. As the result, MVDepthNet-64 estimates the depth maps at 25 fps (0.04 s) when the input image size is 320 $\times$ 240. However, the accuracy is not as high as other state-of-the-art methods.

DPSNet [9] also introduced a differentiable warping module for end-to-end training, and achieved a state-of-the-art

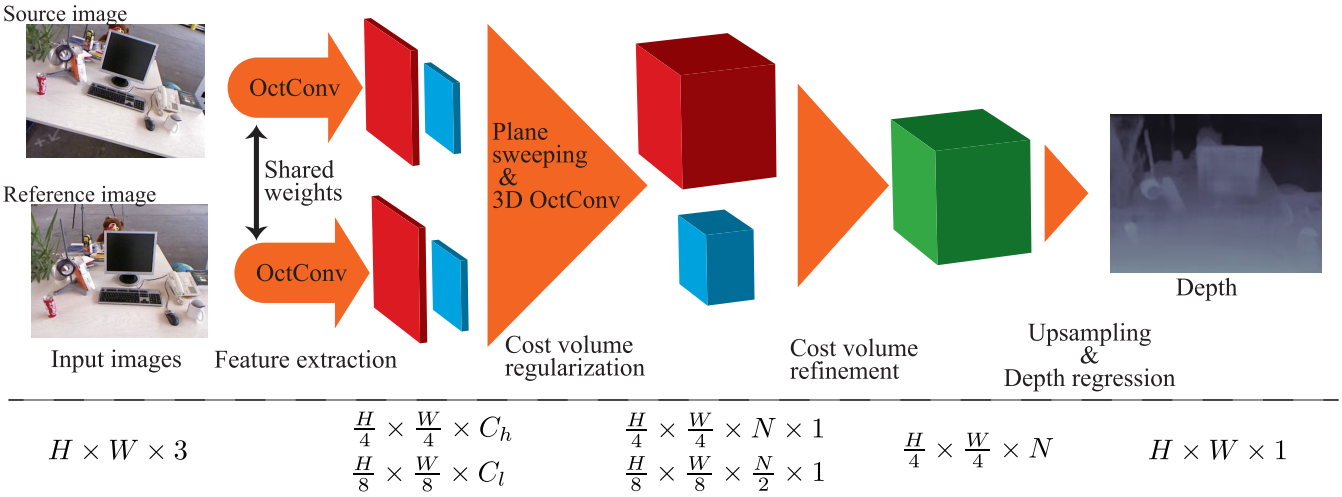

FIGURE 1. Overview of our proposed network. The input consists of the reference and source images and the relative camera pose used for plane-sweeping, and the output is a depth aligned with the reference image. The red box indicates high spatial frequency features, the blue box indicates low spatial frequency features, and the green box indicates a mixture of high and low spatial frequency features. The row below shows the shapes of the features.

depth estimation accuracy. The authors proposed a context-aware refinement to accurately estimate depth maps even in regions with poor texture. As this method requires less than a second (0.5 s) to estimate depth maps while achieving a state-of-the-art accuracy, we chose DPSNet as base model for our proposed model. However, this requires a significant amount of memory during training, which makes it difficult to apply in some cases. We also tackle the memory consumption problem in this work.

III. METHOD

A. OVERVIEW

The overview of our proposed method is similar to previous learning-based plane-sweeping methods, especially [9]. The biggest difference is that we process high and low spatial frequency features separately, both for the image features and cost volume. The overview is presented in Fig. 1.

First, high and low spatial frequency features are extracted from the reference and source images using a feature extraction module.

Second, cost volumes are generated from the extracted features by plane-sweeping, and these are regularized. Here, two kinds of cost volume are generated, from the high and low spatial frequency features, respectively.

Third, these cost volumes are refined and integrated into a single final cost volume.

Finally, the depth is obtained by depth regression and upsampled to the original image dimension. The obtained depth is aligned with the reference image.

B. OCTAVE CONVOLUTION

OctConv [14] was proposed to reduce the spatial redundancy of features, which enables less memory and computational resources to be utilized without degrading accuracy. The difference between the concepts of ConvNet and

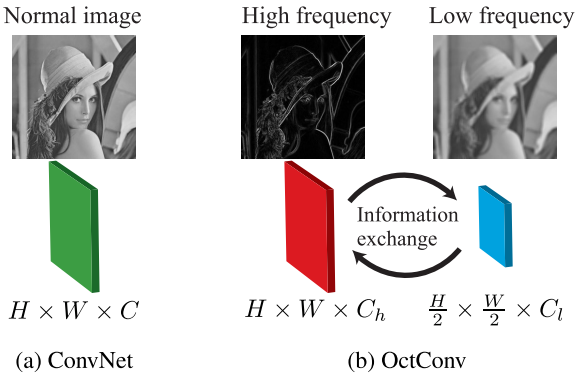

FIGURE 2. Concepts of normal convolution (ConvNet) and octave convolution (OctConv). (a) illustrates the concept of ConvNet. This processes a mixture of high and low spatial frequency features, which constitutes a normal image. (b) illustrates the concept of OctConv. This deals with high and low spatial frequency features separately, but also updates information between them. The row below shows the shapes of the features.

OctConv is illustrated in Fig. 2. ConvNet processes features that are mixtures of high and low spatial frequencies, whereas OctConv processes high and low spatial frequency features separately while exchanging information between them. Here, the key is to only employ a low resolution ($H/2 \times W/2$, as shown in Fig. 2(b)) for the low spatial frequency features, which enables the spatial redundancy to be reduced. OctConv, with inputs $\{X_h, X_l\}$ and outputs $\{Y_h, Y_l\}$, is formulated as follows [14]:

$$Y_h = f(X_h; W_{h\to h}) + \mathsf{upsample}(f(X_l; W_{l\to h})), \quad (1)$$

$$Y_l = f(X_l; W_{l\to l}) + f(\mathsf{pool}(X_h); W_{h\to l}), \quad (2)$$

where $f(X; W)$ represents a convolution with weight parameters W, $\mathsf{upsample}(X)$ indicates an upsampling of X by a factor of 2, and $\mathsf{pool}(X)$ is an average pooling operation

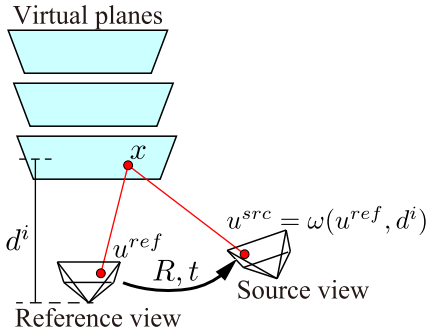

FIGURE 3. Illustration of plane sweeping. Virtual fronto-parallel planes are generated at several distances d^i from the reference image plane, and the features from the reference and source images are projected onto the planes to calculate the costs. R and t are the 3×3 rotation matrix and the three-dimensional translation vector from the reference to source viewpoint, respectively.

with kernel size 2×2 and stride 2. The subscripts h and l represent high and low spatial frequency features, respectively. OctConv has a hyper-parameter α that controls the ratio of low spatial frequency features. For example, $\alpha = 0.75$ implies that 75% of the features are low spatial frequency features. If α is increased, then less memory and computation resources are used, but useful information may be lost. For further details, see the study [14].

C. FEATURE EXTRACTION

Our feature extraction module is based on ResNet-34 [17]. We utilize OctConv instead of ConvNet, except for the first ConvNet, to extract high and low spatial frequency features. Furthermore, an SPP layer [18] is added at the last layer to capture global contextual information, as in PSMNet [7]. The shape of the input image is $H \times W \times 3$, and the shapes of the high and low spatial frequency features F_h and F_l are $H/4 \times W/4 \times C_h$ and $H/8 \times W/8 \times C_l$, respectively. Here, $C_h = C_{all} \times (1 - \alpha)$ and $C_l = C_{all} \times \alpha$, where $C_{all} = 32$ is utilized in this study. The features $\{F_h^{ref}, F_l^{ref}\}$, and $\{F_h, F_l\}$ are extracted from the reference and source images, as shown in Fig. 1, but the both feature extraction modules have shared weight parameters.

D. PLANE SWEEPING

The extracted features are utilized to generate cost volumes by plane sweeping. Two cost volumes are generated: V_h and V_l from the high and low spatial frequency features, respectively. Plane sweeping is performed in the same manner as in traditional methods [4], [5].

In plane sweeping, virtual fronto-parallel planes are generated at several distances from the reference image plane, and the features from the reference and source images are projected onto the planes to calculate the costs as shown in Fig. 3. A voxel of the cost volume from the high spatial frequency features is formulated as follows:

$$V_h(u^{ref}, i) = \mathsf{concat}\left(F_h^{ref}\left(u^{ref}\right), F_h\left(\omega\left(u^{ref}, d_h^i\right)\right)\right), \quad (3)$$

$$\omega\left(u^{ref}, d_h^i\right) = \pi\left(R\pi^{-1}\left(u^{ref}, d_h^i\right) + t\right), \quad (4)$$

where $\mathsf{concat}(\cdot, \cdot)$ is the concatenation operator, $\pi(x)$ projects a 3D point x onto an image plane, and $\pi^{-1}(u, d)$ is the inverse function that projects an image point u at depth d back onto 3D space. We assume the intrinsic parameters of the camera are already known. Furthermore, R and t are the 3×3 rotation matrix and the three-dimensional translation vector from the reference to source viewpoint, respectively. In addition, d_h^i is the distance between the i-th virtual plane and the optical center of the reference image, which will be discussed in Section III-F. Because we use the spatial transformer network [19] for the projection of the features, the whole plane-sweeping process is fully differentiable, which makes it possible to train our model in an end-to-end manner.

A voxel of the cost volume from the high spatial frequency features V_l is also generated in the same manner as in (3) and (4), except that the low spatial frequency features are used and the numbers and distances of virtual planes are different.

E. NUMBER OF VIRTUAL PLANES IN PLANE SWEEPING

The more virtual planes are generated at different distances, the more likely it is to capture fine structures (depths), however, the more computation time is required [12]. To deal with the trade-off between the computation time and accuracy, we focus on reducing the spatial redundancy in a manner inspired by OctConv [14]. Because low spatial frequency features do not have fine textures, fewer virtual planes are required for these than for the high spatial frequency features.

We set the number of virtual planes N_l for low spatial frequency features to half the number N_h for the high spatial frequency features, i.e., $N_l = N_h/2$. Furthermore, this enables 3D OctConv to be employed to regularize the cost volumes. 3D OctConv is the 3D version of OctConv, which uses 3D ConvNet instead of 2D ConvNet in (3) and (4) to process voxels, and it requires the cost volume V_l to have half the resolution of V_h. Thus, the shapes of V_h and V_l are $H/4 \times W/4 \times N_h \times 2\ C_h$ and $H/8 \times W/8 \times N_h/2 \times 2\ C_l$, respectively. We set $N_h = 64$ in this study.

F. REGULARIZATION OF THE COST VOLUMES

1) DEALING WITH MISALIGNMENT

To utilize 3D OctConv to regularize the cost volumes, the distances of the virtual planes must be chosen carefully to deal with the misalignment between the high and low spatial frequency features, as discussed in OctConv [14]. As shown in (2), an average pooling operation with kernel size 2×2 and a stride of 2 are utilized (for 3D OctConv a kernel size $2 \times 2 \times 2$ is adopted) to exchange information between the high and low spatial frequency features. Therefore, the distances d_h^i and d_l^i between the i-th virtual planes and the optical center of the reference image should be formulated as follows, so that after the pooling operation V_h aligns with V_l (Fig. 4):

$$d_h^i = \frac{N_h d_{min}}{i}, \quad i \in \{1, 2, \ldots, N_h\}, \quad (5)$$

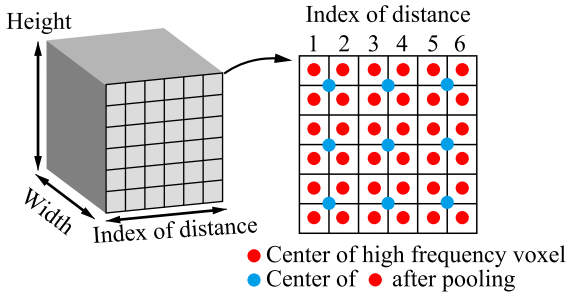

FIGURE 4. Cost volume alignment for 3D OctConv. Left: Illustration of the cost volume. Right: Red points indicate the centers of high spatial frequency voxels. Blue points represent the centers after the pooling operations. The blue points should align with the low spatial frequency voxels.

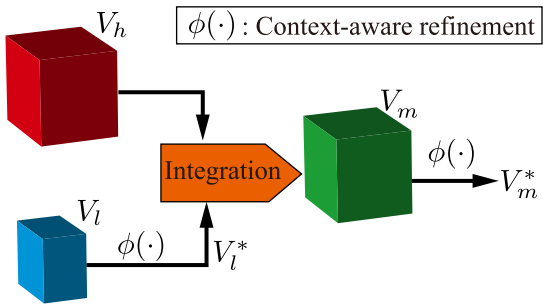

FIGURE 5. Overview of refinement. The red and blue boxes represent the cost volumes V_h and V_l, respectively. The green box represents the integrated cost volume V_m. A superscript asterisk indicates processing by context-aware refinement.

$$d_l^i = \frac{N_h d_{min}}{2i - 0.5}, \quad i \in \{1, 2, \ldots, N_h/2\}, \tag{6}$$

where d_{min} is the minimum distance between the virtual plane and the optical center of the reference image, and $d_{min} = 0.5$ is adopted in this study. As shown in Fig. 4, if the indices of the distance are $\{1, 2, 3, 4, 5, 6\}$, then the indices after the pooling operation will be $\{1.5, 3.5, 5.5\}$. Therefore, formulating d_h^i and d_l^i as in (5) and (6) solves the misalignment problems in 3D OctConv. It should be noted that the row and column directions of the cost volumes are already aligned.

2) REGULARIZATION BY 3D OCTCONV

The regularization of the cost volumes V_h and V_l is performed using multiple 3D OctConv blocks with kernel size $2 \times 2 \times 2$ with skip connections, in reference to DPSNet [9]. The shapes of V_h and V_l before regularization are $H/4 \times W/4 \times N_h \times 2\,C_h$ and $H/8 \times W/8 \times N_h/2 \times 2\,C_l$, respectively, and these become $H/4 \times W/4 \times N_h \times 1$ and $H/8 \times W/8 \times N_h/2 \times 1$ after regularization.

In the case that more than two view images are available, multiple cost volumes are averaged.

G. COST VOLUME REFINEMENT

An overview of the cost volume refinement process is presented in Fig. 5. We employ the context-aware refinement approach proposed in DPSNet [9]. The low spatial frequency cost volume V_l is refined using context-aware refinement, and V_l^* is first obtained. Furthermore, V_h and V_l^* are integrated by the integration module, which will be described in Section III-H. Finally, the integrated cost volume V_m is refined using context-aware refinement, and the final cost volume V_m^* is generated.

We use five dilated OctConv layers in the context-aware refinement with receptive fields (1,2,4,1,1), instead of seven dilated ConvNet layers, to improve the computational time and decrease memory consumption. Here, V_m and V_m^* have the same shape $H/4 \times W/4 \times N_h$, and these are utilized for depth regression, as described in Section III-I. In addition, both are used in the loss function, which is described in Section III-J.

H. INTEGRATION OF THE COST VOLUMES

The cost volumes V_h and V_l^* are integrated using our proposed module to obtain V_m. Because V_l^* is generated from the low spatial frequency features, it includes information on the rough structures of scenes, whereas V_l^* is generated from high spatial frequency features, and therefore includes fine structures (e.g., edges or small depth changes). The latter also contains some noise, which may degrade the accuracy of the estimated depth.

Here, V_h and V_l^* are integrated by the weighted average in the proposed integration module. However, the weight factor is different depending on the location. For instance, on the one hand a higher weight should be given to V_h in some regions that contain edges or fine structures. On the other hand, in some areas a higher weight should be given to V_l^* to reduce the negative effects of noise.

To estimate the pixel-wise weight factor, we introduce an SE-based [15] attention mechanism, which is commonly used for estimating the important regions in features. The SE module was originally proposed for image classification tasks. As it only estimates the channel-wise importance and does not consider the pixel-wise importance, it is not suitable for pixel-wise estimation tasks. Therefore, a spatial SE (sSE) module was proposed to estimate the pixel-wise importance, and the performance improvement has been demonstrated for semantic segmentation tasks [20].

Our integration module is similar to the sSE module, but it accepts multiple inputs to deal with the two cost volumes, as shown in Fig. 6. In Fig. 6, the blue and red boxes represent V_h and V_l^*, respectively. The integrated cost volume V_m at the location u is formulated as follows:

$$V_m(u) = \lambda V_h(u) + (1 - \lambda)\,\mathsf{upsample}(V_l^*)(u), \tag{7}$$

where $\mathsf{upsample}(X)$ represents an upsampling of X by a factor of 2, and $\{\lambda \in \mathbb{R} \mid 0 \leq \lambda \leq 1\}$ is the weight factor.

In fact, λ is the pixel-wise weight factor, which has the shape $H/4 \times W/4 \times 1$. Thus, the value is different depending on the location. Here, λ is formulated as follows:

$$\lambda = \sigma\left(g_{1\times1}^1(S)\right), \tag{8}$$

$$S = \mathsf{concat}\left(\hat{g}_{1\times1}^{\frac{N_h}{8}}(V_h),\ \hat{g}_{1\times1}^{\frac{N_h}{8}}\left(\mathsf{upsample}(V_l^*)\right)\right), \tag{9}$$

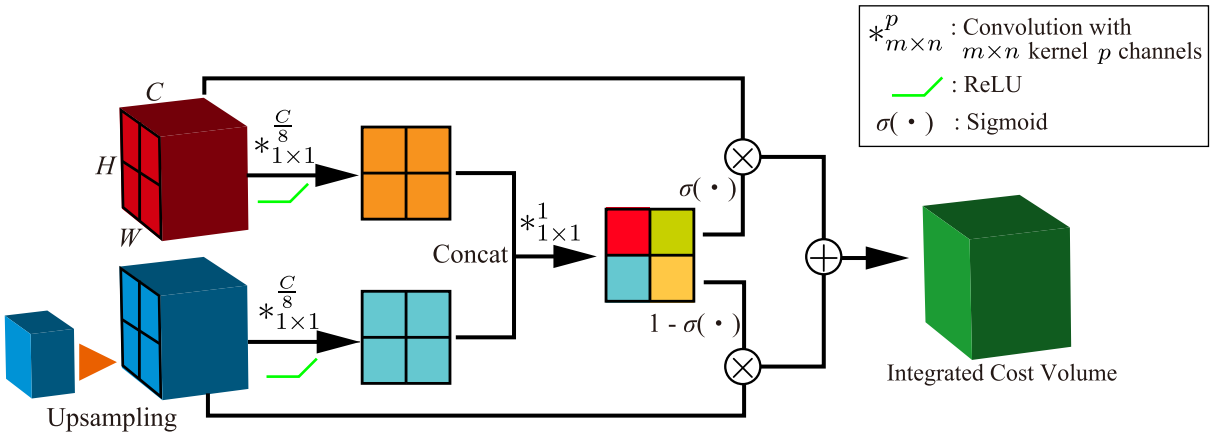

FIGURE 6. Integration module. The blue and red boxes represent V_h and V_l^*, respectively, and these are integrated by a pixel-wise "squeeze-and-excitation"-based attention mechanism. Finally, the integrated cost volume V_m is obtained, represented by the green box.

where $g^c_{1\times1}(\cdot)$ indicates a convolution with kernel size 1×1 and the number of output channels c, and the hat on $\hat{g}^c_{1\times1}(\cdot)$ indicates that the ReLU activation is used at the last layer. In addition, $\sigma(\cdot)$ is the sigmoid function.

I. DEPTH REGRESSION

We use the depth regression method proposed in GC-Net [6] to estimate the continuous depth map from the cost volumes. The depth at location u is calculated from the cost volume V_m^* as follows:

$$\hat{D}^*(u) = \frac{N_h d_{min}}{\sum_{i=1}^{N_h} i\sigma\left(V_m^*(u,\cdot)\right)_i}, \tag{10}$$

where $\sigma(\cdot)_i$ is a softmax operation, formulated as

$$\sigma\left(V_m^*(u,\cdot)\right)_i = \frac{\exp\left(V_m^*(u,i)\right)}{\sum_{j=1}^{N}\exp\left(V_m^*(u,j)\right)}. \tag{11}$$

The cost volume V_m^* is upsampled to the original image dimension ($H \times W \times N_h$) before depth regression. We find that upsampling the cost volume before depth regression instead of upsampling the depth map yields a smoother depth map. Here, trilinear interpolation is utilized in the upsampling.

Only the depth map $\hat{D}^*$ from the refined cost volume V_m^* is used for the inference, however, the depth map $\hat{D}$ from the unrefined cost volume V_m is also used to calculate the loss during training. We find that adding $\hat{D}$ along with $\hat{D}^*$ in the loss function helps to avoid overfitting. The depth map $\hat{D}$ is generated from V_m in the same manner as the depth map $\hat{D}^*$. The loss function will be explained in Section III-J.

J. LOSS FUNCTION

The loss function formulated below is used to minimize the errors between the predicted depths $\hat{D}^*$ and $\hat{D}$ and the ground truth D:

$$L(\hat{D}^*, \hat{D}, D) = L_{depth}(\hat{D}^*, D) + \gamma L_{depth}(\hat{D}, D), \tag{12}$$

where L_{depth} is formulated as

$$L_{depth}(D_1, D_2) = \frac{1}{|\Omega|}\sum_{u\in\Omega} f_\delta\left(D_1(u), D_2(u)\right), \tag{13}$$

Here, f_δ is the Huber loss [21] with $\delta = 1$:

$$f_\delta(z_1, z_2) = \begin{cases} \frac{1}{2}(z_1 - z_2)^2 & \text{if } |z_1 - z_2| \le \delta \\ \delta|z_1 - z_2| - \frac{1}{2}\delta^2 & \text{otherwise} \end{cases}. \tag{14}$$

Furthermore, $\Omega \subset \mathbb{R}^2$ is the image domain, which includes the valid ground truth values, and γ is a weight factor to balance their importance.

IV. EXPERIMENTS

A. DATASETS

We utilized SUN3D [22], RGB-D SLAM [23], MVS [24]–[27], and Scenes11 [28], [29] for training and evaluation. These contain images, depths, and camera poses. They include various indoor and outdoor scenes. Scenes11 is a synthetic dataset, and the others were collected from the real world. The datasets were split into training dataset and test datasets in the same manner as in a study [28].

We used 168,357 image pairs for training, 19,854 pairs from the training dataset for validation, and 708 image pairs from the test dataset for evaluation. The model with the lowest validation error during training was selected as the best model and evaluated. The absolute relative difference, described in Section III, was used to determine the best model. In addition, only ground truth depths from 0.5 m to 10 m were used for the evaluation.

The ETH3D dataset [30] was used for further evaluation. This dataset was only used for evaluation, not for training. ETH3D is a benchmark for multi-view stereo, and contains indoor and outdoor scenes. The ETH3D dataset consists of several datasets, and high-resolution multi-view dataset was used in this study. While the test datasets of SUN3D, RGB-D SLAM, MVS, and Scenes11 only have two view images,

the ETH3D dataset has more than two view images. Thus, the evaluation of the depth estimation from more than two view images is possible on this dataset. Because some depth values of the ground truth depth maps are smaller than 0.5 m ($= d_{min}$), the relative camera poses were adjusted based on the minimum values of the ground truth depth maps.

B. TRAINING DETAILS

We used the PyTorch framework to implement the proposed network. The training was conducted in an end-to-end fashion on two NVIDIA Tesla P100 GPUs with 16 GB of memory. We trained the network for 210,000 iterations with a batch size of 16. This required approximately five–seven days, depending on α. During training, Adam [31] was used as the optimizer and the learning rate was set to 3e-4 for first 160,000 iterations and 6e-5 for the remainder. The hyper-parameters β_1 and β_2 in Adam were set to 0.9 and 0.999, respectively. In addition, the weight factor $\gamma = 0.7$ was adopted for the loss function in (12).

C. DATA AUGMENTATION

Random cropping and zooming were applied to the datasets, and these were resized to 320 × 240 to increase the training speed. The original image size (640 × 480) was used for the evaluation and testing.

Some datasets were collected by cameras (e.g., Kinect) that automatically control the exposure time based on the illumination of a scene. Therefore, it cannot be assumed that the same 3D point projected in several images has the same intensity (photometric consistency). To remedy this, color jittering was also randomly applied to each image to yield intensity differences between the reference and source images during training. We also added Gaussian noise, changing the image contrast, multiplying images by random values, and adding random values to images. The effect of color jittering will be illustrated in Section V-A.

D. METRICS

The results were evaluated using the depth estimation metrics [32] of the root-mean-square error (RMSE), log RMSE, absolute difference (Abs Diff), absolute relative difference (Abs Rel), squared relative difference (Sq Rel), and threshold (Threshold). These are expressed by the following equations:

$$Abs\ Diff\ :\ \frac{1}{|T|}\sum_{y\in T}\left|y_i-\hat{y}_i\right|, \tag{15}$$

$$Abs\ Rel\ :\ \frac{1}{|T|}\sum_{y\in T}\frac{\left|y_i-\hat{y}_i\right|}{\hat{y}_i}, \tag{16}$$

$$Sq\ Rel\ :\ \frac{1}{|T|}\sum_{y\in T}\frac{\left|y_i-\hat{y}_i\right|^2}{\hat{y}_i}, \tag{17}$$

$$RMSE\ :\ \sqrt{\frac{1}{|T|}\sum_{y\in T}\left(y_i-\hat{y}_i\right)^2}, \tag{18}$$

TABLE 1. Ablation study on our proposed model. The best scores are indicated in bold.

Configuration			Error (smaller is better)				
SWEEP	INTG	COLOR	Abs Rel	Abs Diff	Sq Rel	RMS	log RMS
✓			0.097	0.287	0.139	0.509	0.169
	✓		0.097	0.282	**0.126**	0.502	**0.165**
✓	✓		0.095	0.277	0.129	0.500	0.166
✓	✓	✓	**0.094**	**0.273**	0.131	**0.492**	**0.165**

TABLE 2. Contribution of color jittering. The best scores are indicated in bold.

Dataset	COLOR	Error (smaller is better)				
		Abs Rel	Abs Diff	Sq Rel	RMS	log RMS
MVS	✓	**0.074**	**0.193**	**0.071**	**0.416**	**0.151**
	NO	0.076	0.206	0.078	0.436	0.163
SUN3D	✓	**0.130**	**0.298**	**0.087**	**0.388**	**0.175**
	NO	0.138	0.314	0.100	0.411	0.188
RGB-D SLAM	✓	**0.137**	**0.488**	0.241	**0.647**	0.229
	NO	0.139	0.489	**0.221**	0.664	**0.223**
Scenes11	✓	0.055	0.165	0.120	0.498	0.125
	NO	**0.052**	**0.158**	**0.115**	**0.484**	**0.119**

$$log\ RMSE\ :\ \sqrt{\frac{1}{|T|}\sum_{y\in T}\left(\log y_i-\log\hat{y}_i\right)^2}, \tag{19}$$

$$Threshold: \%\ of\ y_i\ s.t.\ \max\left(\frac{y_i}{\hat{y}_i},\frac{\hat{y}_i}{y_i}\right)=\delta<Threshold, \tag{20}$$

where y_i and $\hat{y}_i$ are the ground truth and prediction value, respectively.

V. RESULTS

A. ABLATION STUDY

An ablation study was performed to assess the contributions of our proposed components, as shown in Table 1. The ratio of the low spatial frequency features $\alpha = 0.75$ was chosen for the ablation study, i.e., $C_h = 8$ and $C_l = 24$. The results were obtained by the networks trained for 160,000 iterations.

The checkmark in the SWEEP column indicates that our proposed plane-sweeping module from Section. III-F was used. Otherwise, a simple version of the plane-sweeping module was utilized, formulated as follows:

$$d'^{i}_{h}=\frac{N_h d_{min}}{i},\quad i\in\{1,2,\ldots,N_h\}, \tag{21}$$

$$d'^{i}_{l}=\frac{N_h d_{min}}{2i},\quad i\in\{1,2,\ldots,N_h/2\}. \tag{22}$$

The checkmark in the INTG column indicates that our proposed integration module from Section III-H was used. Otherwise, an addition operator was used to integrate the two cost volumes. The checkmark in the COLOR column indicates that the color jittering process explained in Section IV-C was used for data augmentation.

As shown in Table 1, our proposed integration module contributed to the depth estimation accuracy, as the errors were decreased, whereas the importance of the plane-sweeping module was minor. We believe that that is because the integration module and context-aware refinement method have

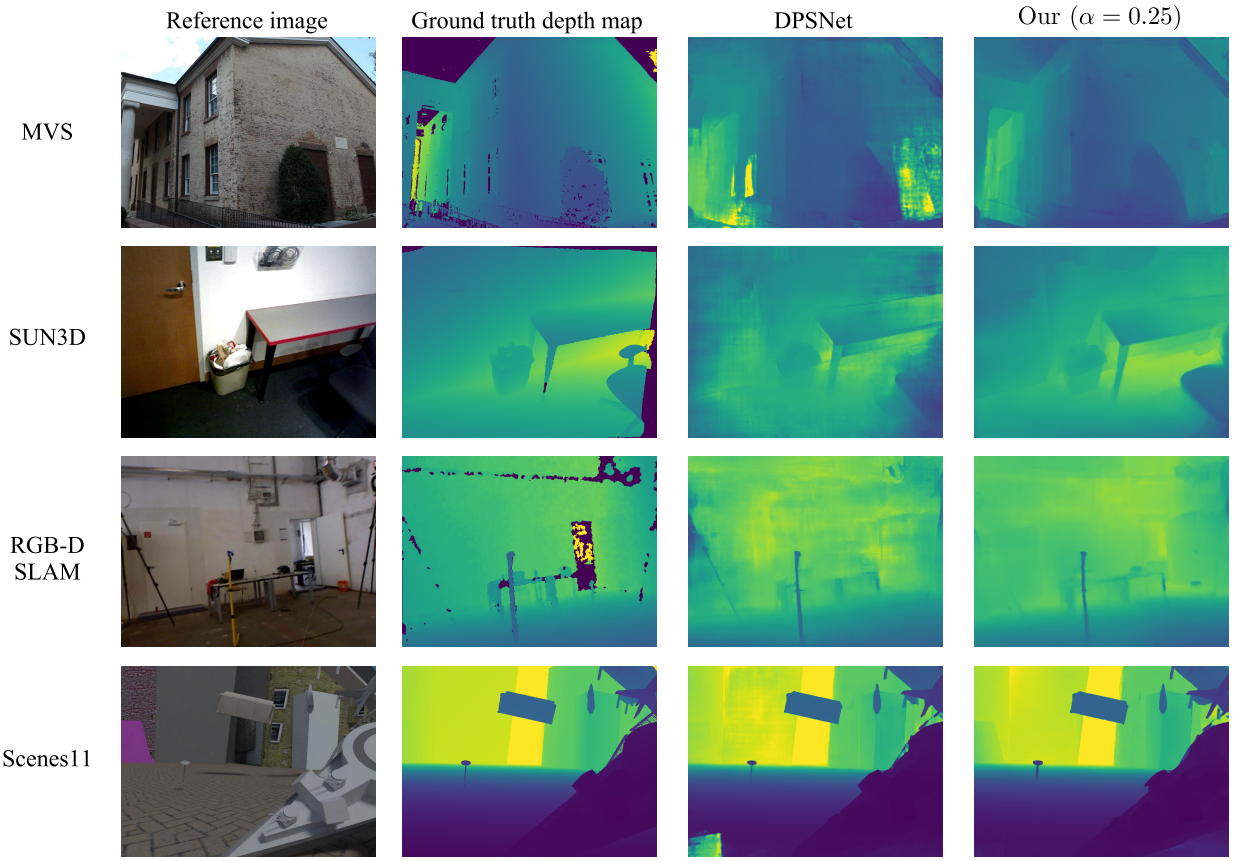

FIGURE 7. Examples of the estimated depth maps on each dataset. The names of the datasets are shown on the left. The four columns from left to right correspond to the reference images, the ground truths of the depth maps, the depth maps estimated by DPSNet, and the depth maps estimated by our model ($\alpha = 0.25$).

TABLE 3. Comparison of different α values in our proposed method. The results for DPSNet [9] are also shown, for a better comparison. The channel number of the high spatial frequency features is written in brackets in the first column, along with the parameter α.

	Error (smaller is better)					Threshold (bigger is better)			(smaller is better)	
	Abs Rel	Abs Diff	Sq Rel	RMS	log RMS	$\delta < 1.25$	$\delta < 1.25^2$	$\delta < 1.25^3$	Speed [s]	Memory [GB]
α=0.25 (24)	**0.0873**	**0.2518**	**0.1111**	**0.4512**	**0.1522**	0.8999	**0.9608**	0.9780	0.40	27.6
α=0.5 (16)	0.0875	0.2521	0.1180	0.4555	0.1548	0.9028	0.9602	0.9774	0.34	21.7
α=0.75 (8)	0.0883	0.2560	0.1182	0.4674	0.1550	0.9033	0.9603	0.9776	0.29	14.3
α=0.875 (4)	0.0896	0.2566	0.1221	0.4730	0.1547	**0.9040**	0.9605	**0.9789**	0.26	12.1
α=0.9375 (2)	0.0953	0.2841	0.1353	0.5033	0.1667	0.8914	0.9531	0.9725	**0.25**	**10.6**
DPSNet	0.1012	0.2898	0.1424	0.5128	0.1717	0.8757	0.9481	0.9734	0.49	43.1

some flexibility to compensate for the misalignment between the two cost volumes.

More detailed results on the influence of color are presented in Table 2. As can be observed from Table 2, color jittering has a negative effect on the result on Scene11. This is because this is a synthetic dataset, and the photometric consistency is not compromised. In this study, we chose to adopt color jittering although it only leads to a slight improvement.

B. COMPARISON OF DIFFERENT α VALUES

A comparison of different values for the α parameter in our proposed method is presented in Table 3. The results on DPSNet are also presented in Table 3, for a better comparison. In addition to the accuracy, the inference speed (Speed) and amount of GPU memory required for training (Memory) are shown in Table 3. The detailed inference time is shown in Table 4. Our proposed model consists of four parts: (1) Feature extraction (EXTRACT), (2) Plane sweeping & Cost volume regularization (SWEEPING), (3) Cost volume refinement (REFINE), and (4) Upsampling & Depth regression, as shown in Fig. 1. Therefore, the computation time on each

TABLE 4. Inference speed on each part of our proposed method. The results of DPSNet [9] are also shown, for a better comparison. The inference speed of "Upsampling & Depth regression" is not shown here because it takes less than 0.01 s.

	Speed [s]			
Model	EXTRACT	SWEEPING	REFINE	Total
Our (α=0.25)	0.01	0.30	0.09	0.40
Our (α=0.50)	0.01	0.24	0.09	0.34
Our (α=0.75)	0.01	0.18	0.09	0.29
Our (α=0.875)	0.01	0.16	0.09	0.26
Our (α=0.9375)	0.01	0.14	0.09	0.25
DPSNet	0.02	0.31	0.16	0.49

TABLE 5. Detailed comparison of different α values for our proposed method on the MVS, SUN3D, RGB-D SLAM, and Scenes11 datasets. The results of DPSNet [9] are also shown, for a better comparison. The best scores are indicated in bold, and the second-best are underlined.

Dataset	Model	Error (smaller is better)					Threshold (bigger is better)		
		Abs Rel	Abs Diff	Sq Rel	RMS	log RMS	$\delta < 1.25$	$\delta < 1.25^2$	$\delta < 1.25^3$
MVS	Our (α=0.25)	**0.0688**	**0.1699**	**0.0571**	**0.3819**	**0.1446**	0.9054	**0.9648**	**0.9868**
	Our (α=0.5)	0.0744	0.1902	0.0741	0.4208	0.1595	0.8917	0.9527	0.9803
	Our (α=0.75)	0.0696	0.1820	0.0662	0.4039	0.1467	**0.9079**	0.9641	0.9846
	Our (α=0.875)	0.0729	0.1904	0.0691	0.4139	0.1519	0.9012	0.9602	0.9846
	Our (α=0.9375)	0.0928	0.2878	0.1252	0.5239	0.1853	0.8636	0.9354	0.9660
	DPSNet	0.0820	0.2021	0.0978	0.4452	0.1607	0.8920	0.9526	0.9809
SUN3D	Our (α=0.25)	0.1283	0.2975	**0.0832**	0.3886	0.1731	0.8220	0.9517	0.9828
	Our (α=0.5)	0.1298	0.2979	0.0918	0.3937	0.1769	0.8280	0.9485	0.9812
	Our (α=0.75)	0.1281	0.2990	0.0852	0.3888	0.1747	0.8238	0.9446	0.9800
	Our (α=0.875)	**0.1271**	**0.2911**	0.0838	**0.3808**	**0.1706**	**0.8355**	**0.9520**	**0.9829**
	Our (α=0.9375)	0.1293	0.2970	0.0865	0.3857	0.1748	0.8286	0.9446	0.9792
	DPSNet	0.1485	0.3387	0.1182	0.4531	0.1976	0.7845	0.9310	0.9786
RGB-D SLAM	Our (α=0.25)	0.1310	0.4652	0.2226	0.6147	0.2131	0.8573	0.9230	0.9412
	Our (α=0.5)	**0.1252**	0.4525	0.2193	0.5941	0.2062	**0.8722**	**0.9323**	0.9456
	Our (α=0.75)	0.1270	0.4494	0.2114	0.6029	0.2060	0.8718	0.9315	0.9472
	Our (α=0.875)	0.1256	**0.4362**	**0.1953**	**0.5858**	**0.1958**	0.8718	0.9310	**0.9511**
	Our (α=0.9375)	0.1307	0.4626	0.2208	0.6109	0.2148	0.8571	0.9248	0.9421
	DPSNet	0.1517	0.5345	0.2530	0.6995	0.2436	0.8092	0.9003	0.9326
Scenes11	Our (α=0.25)	**0.0442**	0.1324	**0.0865**	**0.4233**	**0.1052**	0.9717	0.9880	**0.9935**
	Our (α=0.5)	0.0446	**0.1305**	0.0934	0.4250	0.1065	**0.9739**	**0.9887**	**0.9935**
	Our (α=0.75)	0.0492	0.1468	0.1072	0.4639	0.1152	0.9698	0.9860	0.9914
	Our (α=0.875)	0.0527	0.1570	0.1273	0.4900	0.1206	0.9677	0.9844	0.9908
	Our (α=0.9375)	0.0534	0.1626	0.1172	0.4981	0.1222	0.9659	0.9850	0.9907
	DPSNet	0.0502	0.1521	0.1113	0.4679	0.1165	0.9651	0.9863	0.9918

part during inference is shown in Table 4. It should be noted that the computation time on Upsampling & Depth regression is not included in Table 4 because it takes less than 0.01 s. A single NVIDIA GeForce RTX 2080 Ti GPU was used to evaluate the inference speed, whereas multiple NVIDIA Tesla P100 GPUs with 16 GB of memory (one for our model with $\{\alpha = 0.75, 0.875, 0.9375\}$, two for our model with $\{\alpha = 0.25, 0.5\}$, and four for DPSNet) were used to measure the required GPU memory.

As can be observed from Table 3, our model ($\alpha = 0.25$) achieved the best scores, while the inference speed was faster and the required memory was lower compared to DPSNet. Even our fastest model ($\alpha = 0.9375$), which has a lower accuracy, achieved higher scores than DPSNet except for the Threshold $\delta < 1.25^3$. This demonstrates the effectiveness of our proposed method. As can be observed from Table 4, our proposed method contributes to reduce computation time on REFINE regardless of α and SWEEPING when α is large. It should be noted that because our proposed model requires considerably less memory, our model ($\alpha = 0.9375$) can be trained on even a single consumer GPU (e.g., NVIDIA GeForce RTX 2080 Ti with 11 GB of memory).

A detailed comparison of results on each dataset is presented in Table 5. As shown in Table 5, our models with higher α values (more low spatial frequency features) achieved higher scores on the SUN3D and RGB-D SLAM datasets, while our models with lower α values (more high spatial frequency features) achieved higher scores on the MVS and Scenes11 datasets. We believe that the difference in performances is a result of scene characteristics. The SUN3D and RGB-D SLAM datasets include indoor scenes, and so they contain many textureless regions (e.g., walls and floors). Therefore, low frequency features are important. On the other hand, the MVS dataset contains outdoor scenes, and Scenes11 contains synthetic scenes with strong edges. Thus, high frequency features are effective.

Examples of the estimated depth maps for each dataset are presented in Fig. 7. It can be observed from Fig. 7 that the depth maps estimated by DPSNet contain some artifacts in wide textureless regions (e.g., walls and floors in SUN3D and RGB-D SLAM) and reflection regions (e.g., glasses in MVS), whereas our model succeeded in estimating the depth maps even in these difficult regions.

C. COMPARISON WITH PREVIOUS METHODS ON THE ETH3D DATASET

A further comparison with previous methods was conducted on the ETH3D dataset [30], as shown in Table 6. Here, DPSNet [9], DeepMVS [10], MVSNet [8], and R-MVSNet [11] were chosen as the previous methods. We resized the images to 810 × 540 for our model and DPSNet, and 864 × 576 for MVSNet and R-MVSNet. Because the ETH3D dataset provides more than two view images, the number of images for depth estimation needs to be determined. Three view images were utilized for our model, MVSNet, and R-MVSNet, as this produced the best performance. Four view images were used for DPSNet according to their paper [9]. For MVSNet and R-MVSNet, the depth map refinement module was disabled because this was not publicly available, and pixels with probability lower than 0.3 and 0.15 were discarded as outliers, respectively. We utilized the results for DeepMVS reported on the project webpage. [1]

[1] https://phuang17.github.io/DeepMVS/index.html

TABLE 6. Comparison of our model with previous methods on the ETH3D dataset. The best scores are indicated in bold.

	Error (smaller is better)					Threshold (bigger is better)		
Model	Abs Rel	Abs Diff	Sq Rel	RMS	log RMS	$\delta < 1.25$	$\delta < 1.25^2$	$\delta < 1.25^3$
Our (α=0.25)	0.0933	0.6567	0.2710	1.3683	0.1902	0.8766	0.9390	0.9628
Our (α=0.50)	0.0888	**0.5654**	0.2376	1.2252	0.1759	0.8902	0.9479	0.9685
Our (α=0.75)	**0.0848**	0.5791	**0.2351**	1.2500	**0.1708**	**0.8959**	**0.9507**	**0.9702**
Our (α=0.875)	0.0886	0.5831	0.2399	1.2477	0.1754	0.8937	0.9480	0.9682
Our (α=0.9375)	0.0942	0.6626	0.2646	1.3707	0.1874	0.8818	0.9441	0.9639
DPSNet	0.0952	0.5730	0.2581	**1.2088**	0.1889	0.8738	0.9417	0.9663
DeepMVS	0.1903	0.7790	2.2761	1.9460	0.2500	0.8590	0.9128	0.9439
MVSNet	0.6234	2.1830	14.9020	4.0080	0.4119	0.8065	0.8514	0.8787

As shown in Table 6, our model ($\alpha = 0.75$) achieved the best performance in terms of Abs Rel, Sq Rel, log RMS, and Threshold. Because the ETH3D dataset contains both indoor and outdoor scenes, both high and low spatial frequency features appear to be important.

The results in Tables 5 and 6 suggest that varying α based on scenes might improve the depth estimation accuracy. This is left as a topic of future work.

VI. DISCUSSION

Our original purpose was reducing the memory consumption and computational cost without degrading the accuracy by dividing the cost volume into two parts: a high-frequency part and a low-frequency part. However, our proposed method achieved better depth estimation accuracies than previous methods. As can be seen in Fig. 7, our proposed method can handle wide textureless regions (e.g., walls and floors in SUN3D and RGB-D SLAM). Therefore, it is suggested that dividing the cost volume into two parts help not only to reduce the memory consumption and computational cost but also to exploit global contextual information.

As can be observed from Table 6, MVSNet [8] shows poor performance compare to others, despite that MVSNet achieved state-of-the-art depth estimation accuracies on the DTU [33] and "Tanks and Temples" [34] datasets in their paper. We investigated the DTU and "Tanks and Temples" datasets, and it seems that images were taken densely (small baseline and a lot of overlap regions between images) in the DTU and "Tanks and Temples" datasets whereas images were taken sparsely (wider baseline and smaller overlap regions between images) in the ETH3D dataset [30] we used in this paper. DPSNet [9] is designed to exploit global contextual information by using context-aware refinement. Our method is also capable of exploiting global contextual information. It is suggested that global contextual information help DPSNet and our method produce good depth estimation results even though overlap regions between images are small. On the other hand, MVSNet produces poor depth estimations when overlap regions between images are small.

VII. CONCLUSION

In this paper, we proposed OctDPSNet to reduce the spatial redundancy in learning-based plane-sweeping stereo. Two cost volumes, one with a lower resolution, were generated from the high and low spatial frequency features, respectively, using the proposed plane-sweeping approach considering the misalignment problem. Furthermore, the two cost volumes were integrated using our proposed integration module, which utilizes an attention mechanism. The evaluation results demonstrate that our proposed OctDPSNet outperforms previous methods on five datasets while drastically reducing the required memory and computational time.

In future work, further improvement is expected by varying the ratio of low spatial frequency features α based on the scene characteristics.

ACKNOWLEDGMENT

A part of this research was conducted using the SGI Rackable C2112-4GP3/C1102-GP8 (Reedbush-U/H/L) in the Information Technology Center, The University of Tokyo. The authors would like to thank the members of Intelligent Construction Systems Laboratory, the University of Tokyo, for useful suggestions, especially Mr. Shingo Yamamoto and Mr. Takumi Chiba from Fujita Corporation and Dr. Kazuhiro Chayama from KOKANKYO Engineering Corporation.

REFERENCES

[1] D. Marr and T. Poggio, "Cooperative computation of stereo disparity," *Science*, vol. 194, no. 4262, pp. 283–287, 1976.

[2] H. C. Longuet-Higgins, "A computer algorithm for reconstructing a scene from two projections," *Nature*, vol. 293, no. 5828, pp. 133–135, 1981.

[3] M. Okutomi and T. Kanade, "A multiple-baseline stereo," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 15, no. 4, pp. 353–363, Apr. 1993.

[4] R. T. Collins, "A space-sweep approach to true multi-image matching" in *Proc. CVPR IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, Jun. 1996, pp. 358–363.

[5] D. Gallup, J.-M. Frahm, P. Mordohai, Q. Yang, and M. Pollefeys, "Real-time plane-sweeping stereo with multiple sweeping directions," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2007, pp. 1–8.

[6] A. Kendall, H. Martirosyan, S. Dasgupta, P. Henry, R. Kennedy, A. Bachrach, and A. Bry, "End-to-end learning of geometry and context for deep stereo regression," in *Proc. IEEE Int. Conf. Comput. Vis.*, Oct. 2017, pp. 66–75.

[7] J.-R. Chang and Y.-S. Chen, "Pyramid stereo matching network," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 5410–5418.

[8] Y. Yao, Z. Luo, S. Li, T. Fang, and L. Quan, "MVSNET: Depth inference for unstructured multi-view stereo," in *Proc. Eur. Conf. Comput. Vis.*, 2018, pp. 767–783.

[9] S. Im, H.-G. Jeon, S. Lin, and I. S. Kweon, "DPSNet: End-to-end deep plane sweep stereo," in *Proc. Int. Conf. Learn. Represent.*, May 2019, pp. 1–12.

[10] P.-H. Huang, K. Matzen, J. Kopf, N. Ahuja, and J.-B. Huang, "DeepMVS: Learning multi-view stereopsis," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 2821–2830.

[11] Y. Yao, Z. Luo, S. Li, T. Shen, T. Fang, and L. Quan, "Recurrent MVSNet for high-resolution multi-view stereo depth inference," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2019, pp. 5525–5534.
[12] K. Wang and S. Shen, "MVDepthNet: Real-time multiview depth estimation neural network," in *Proc. Int. Conf. 3D Vis. (3DV)*, Sep. 2018, pp. 248–257.
[13] Y. Huang, Y. Cheng, A. Bapna, O. Firat, M. X. Chen, D. Chen, H. Lee, J. Ngiam, Q. V. Le, Y. Wu, and Z. Chen, "GPipe: Efficient training of giant neural networks using pipeline parallelism," Nov. 2018, *arXiv:1811.06965*. [Online]. Available: https://arxiv.org/abs/1811.06965
[14] Y. Chen, H. Fan, B. Xu, Z. Yan, Y. Kalantidis, M. Rohrbach, S. Yan, and J. Feng, "Drop an Octave: Reducing spatial redundancy in convolutional neural networks with octave convolution," Apr. 2019, *arXiv:1904.05049*. [Online]. Available: https://arxiv.org/abs/1904.05049
[15] J. Hu, L. Shen, and G. Sun, "Squeeze-and-excitation networks," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 7132–7141.
[16] P. Krähenbühl and V. Koltun, "Efficient inference in fully connected CRFs with Gaussian edge potentials," in *Proc. Adv. Neural Inf. Process. Syst.*, 2011, pp. 109–117.
[17] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2016, pp. 770–778.
[18] K. He, X. Zhang, S. Ren, and J. Sun, "Spatial pyramid pooling in deep convolutional networks for visual recognition," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 37, no. 9, pp. 1904–1916, Sep. 2015.
[19] M. Jaderberg, K. Simonyan, A. Zisserman, and K. Kavukcuoglu, "Spatial transformer networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2015, pp. 2017–2025.
[20] A. G. Roy, N. Navab, and C. Wachinger, "Recalibrating fully convolutional networks with spatial and channel 'squeeze and excitation' blocks," *IEEE Trans. Med. Imag.*, vol. 38, no. 2, pp. 540–549, Feb. 2019.
[21] P. J. Huber, "Robust estimation of a location parameter," *Ann. Math. Statist.*, vol. 53, no. 1, pp. 73–101, 1964.
[22] J. Xiao, A. Owens, and A. Torralba, "SUN3D: A database of big spaces reconstructed using SfM and object labels," in *Proc. IEEE Int. Conf. Comput. Vis.*, Dec. 2013, pp. 1625–1632.
[23] J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers, "A benchmark for the evaluation of RGB-D SLAM systems," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Oct. 2012, pp. 573–580.
[24] J. L. Schönberger and J.-M. Frahm, "Structure-from-motion revisited," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2016, pp. 4104–4113.
[25] J. L. Schönberger, E. Zheng, J.-M. Frahm, and M. Pollefeys, "Pixelwise view selection for unstructured multi-view stereo," in *Proc. Eur. Conf. Comput. Vis.*, Sep. 2016, pp. 501–518.
[26] S. Fuhrmann, F. Langguth, and M. Goesele, "MVE–a multi-view reconstruction environment," in *Proc. Eurographics Workshop Graph. Cultural Heritage*, Oct. 2014, pp. 11–18.
[27] B. Ummenhofer and T. Brox, "Global, dense multiscale reconstruction for a billion points," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Dec. 2015, pp. 1341–1349.
[28] 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 *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jul. 2017, pp. 5038–5047.
[29] A. X. Chang, T. Funkhouser, L. Guibas, P. Hanrahan, Q. 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," Dec. 2015, *arXiv:1512.03012*. [Online]. Available: https://arxiv.org/abs/1512.03012
[30] T. Schöps, J. L. Schönberger, S. Galliani, T. Sattler, K. Schindler, M. Pollefeys, and A. Geiger, "A multi-view stereo benchmark with high-resolution images and multi-camera videos," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jul. 2017, pp. 2538–2547.
[31] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," Dec. 2014, *arXiv:1412.6980*. [Online]. Available: https://arxiv.org/abs/1412.6980
[32] D. Eigen, C. Puhrsch, and R. Fergus, "Depth map prediction from a single image using a multi-scale deep network," in *Proc. Adv. Neural Inf. Process. Syst.*, 2014, pp. 2366–2374.
[33] H. Aanæs, R. R. Jensen, G. Vogiatzis, E. Tola, and A. B. Dahl, "Large-scale data for multiple-view stereopsis," *Int. J. Comput. Vis.*, vol. 120, no. 2, pp. 153–168, 2016.
[34] A. Knapitsch, J. Park, Q.-Y. Zhou, and V. Koltun, "Tanks and temples: Benchmarking large-scale scene reconstruction," *ACM Trans. Graph.*, vol. 36, no. 4, Jul. 2017, Art. no. 78.

REN KOMATSU received the B.S. degree in engineering from Yokohama National University, Japan, in 2014, and the M.S. degree in engineering from The University of Tokyo, Japan, in 2016, where he is currently pursuing the Ph.D. degree. From 2017 to 2018, he was a Visiting Scholar with the Robotics Institute, Carnegie Mellon University, USA. His research interests include computer vision, visual SLAM, deep learning, and robot teleoperation.

HIROMITSU FUJII received the B.E. and M.E. degrees in engineering and the Ph.D. degree in precision engineering from The University of Tokyo, Tokyo, Japan, in 2005, 2007, and 2016, respectively. From 2007 to 2013, he was an Image Processing Researcher and a Software Engineer with Sony Corporation Ltd. He was a JSPS Research Fellowship, from 2014 to 2016. From 2016 to 2017, and from 2017 to 2018, he was an Assistant Professor and a Lecturer with The University of Tokyo, respectively. He is currently an Associate Professor with the Department of Advanced Robotics, Chiba Institute of Technology. His research interests include signal processing for robotics systems, such as multisensor integration for unmanned teleoperation systems and pattern recognition for the automated diagnoses using robots. He is a member of JSME, SICE, and RSJ.

YUSUKE TAMURA received the B.E., M.E., and Ph.D. degrees in engineering from The University of Tokyo, Tokyo, Japan, in 2003, 2005, and 2008, respectively, where he is currently a Project Associate Professor with the Graduate School of Engineering. From 2006 to 2008, he was a Research Fellow with the Japan Society for the Promotion of Science. He was a Project Researcher with The University of Tokyo, from 2008 to 2012, and an Assistant Professor with Chuo University, Japan, from 2012 to 2015. His research interests include human–robot interaction, remote control technology, and sports engineering. He is a member of JSME and RSJ.

ATSUSHI YAMASHITA (S'00–M'02) received the B.E., M.E., and Ph.D. degrees in precision engineering from The University of Tokyo, Tokyo, Japan, in 1996, 1998, and 2001, respectively, where he has been an Associate Professor with the Department of Precision Engineering, since 2011. From 1998 to 2001, he was a Junior Research Associate with RIKEN (Institute of Physical and Chemical Research). From 2001 to 2008, he was an Assistant Professor with Shizuoka University. From 2006 to 2007, he was a Visiting Associate with the California Institute of Technology. From 2008 to 2011, he was an Associate Professor with Shizuoka University. His research interests include robot vision, image processing, and intelligent sensing for robots. He is a member of ACM, JSPE, RSJ, IEICE, JSME, IEEJ, IPSJ, ITE, and SICE.

HAJIME ASAMA received M.S. and Dr. Eng. degrees from The University of Tokyo (UTokyo), in 1984 and 1989, respectively, where he has been a Professor with the School of Engineering, since 2009. He was with RIKEN, Japan, from 1986 to 2002. He became a Professor of RACE, UTokyo, in 2002. He was a member of Science Council of Japan, from 2014 to 2017, and has been a Council Member, since 2017. He is a Fellow of JSME and RSJ. He received SICE System Integration Division System Integration Award for Academic Achievement, in 2010, and JSME Award (Technical Achievement), in 2018. He was the Vice-President of RSJ, from 2011 to 2012, and an AdCom member of the IEEE Robotics and Automation Society, from 2007 to 2009. He has been the President-elect of IFAC, since 2017, and the President of International Society for Intelligent Autonomous Systems, since 2014.