17th International Conference on Machine Vision Applications (MVA)
Fully Online, July 25–27, 2021.

Bi-directional Recurrent MVSNet for High-resolution Multi-view Stereo

Taku Fujitomi, Seiya Ito, Naoshi Kaneko and Kazuhiko Sumi
Aoyama Gakuin University
5-10-1 Fuchinobe, Chuo-ku, Sagamihara-shi, Kanagawa, Japan
{fujitomi.taku, ito.seiya}@vss.it.aoyama.ac.jp, {kaneko, sumi}@it.aoyama.ac.jp

Abstract

Learning-based multi-view stereo regularizes cost volumes containing spatial information to reduce noise and improve the quality of a depth map. Cost volume regularization using 3D CNNs consumes a large amount of memory, making it difficult to scale up the network architecture. Recent work proposed a cost-volume regularization method that applies 2D convolutional GRUs and significantly reduces memory consumption. However, this uni-directional recurrent processing has a narrower receptive field than 3D CNNs because the regularized cost at a time step does not contain information about future time steps. In this paper, we propose a cost volume regularization method using bi-directional GRUs that expands the receptive field in the depth direction. In our experiments, our proposed method significantly outperforms the conventional methods in several benchmarks while maintaining low memory consumption.

1 Introduction

Multi-view stereo (MVS) aims to recover 3D scenes from multi-view images with known camera parameters. It can handle a wide range of scenes and reconstruct the scene as a point cloud without using depth sensors. Therefore, it is useful to obtain 3D terrain data from aerial photographs and to reconstruct the general outdoor environment in detail. Most of the MVS methods calculate the cost, which is the similarity between the reference image and several neighboring images, and predict the depth map at the viewpoint of the reference image based on the cost. From the depth maps of multiple viewpoints obtained by these processes for all viewpoints, the 3D scene is reconstructed based on the camera position and orientation of each viewpoint as a point cloud.

Learning-based MVS processes the costs sampled on a 3D grid known as the cost volume. The cost volumes are an essential metric for inferring the depth map. However, it often contains noise due to non-Lambertian reflections or occlusions of objects in the image, resulting in an inaccurate depth map. Therefore, the cost volume regularization is applied to reduce noise in the cost volume. The typical method for cost volume regularization is using 3D CNNs, which regularly process 3D information [7]. However, cost volume regularization using 3D CNNs is known to consume a large amount of GPU memory because it requires that the costs of a large number of coordinates be stored in GPU memory at once. Also, when the resolution of the cost volumes is increased to improve the inference accuracy, the memory consumption grows cubically. Therefore, high-resolution cost volume regularization using 3D CNNs is difficult.

To solve this scalability problem, R-MVSNet [8] has been proposed. In this method, the 3D cost volume is divided into hypothesis planes in the depth direction of the reference image viewpoint, and each plane is regularized sequentially from front to back by gated recurrent units (GRUs). This recurrent regularization allows processing per plane, which significantly reduces the GPU memory consumption for inference. Meanwhile, the cost volume regularization using 3D CNNs enables depth prediction considering a wide range of information at all depths. However, the sequential cost regularization by the uni-directional recurrent processing has a narrower receptive field because the regularized cost at a time step includes no information about future time steps.

In this paper, we propose a cost volume regularization using bi-directional GRUs. Our method regularizes cost volume, considering information about future time steps. It performs with a wider receptive field than R-MVSNet [8] and consumes less GPU memory than regularization using 3D CNNs.

2 Related work

In recent years, many learning-based MVS approaches have been based on 3D volumes. SurfaceNet [9] first constructs colored voxel cubes (CVC) for each view, where each voxel contains the RGB values of the input image. The two CVC are then fed into 3D CNNs to obtain the probability on the surface. Instead of directly predicting the 3D model, approaches to recovering the model through depth maps have been proposed. They form cost volumes by plane sweeping and regularize them using CNNs. One of the representative methods is MVSNet [7], which encodes camera geometries into the cost volumes by differential homography warping. Such an approach to in-

fer depth maps by cost regularization using 3D CNNs consumes a large amount of memory because it requires keeping the entire huge cost volumes in memory. Point-MVSNet [10] processes a point cloud reconstructed from a coarse depth map obtained by small-scale MVSNet [7]. The reconstructed point cloud is processed by the PointFlow module that infers residuals between the coarse depth map and an expected dense depth map as a depth map refinement. CasMVSNet [2] proposed a coarse-to-fine approach to improve depth sampling efficiency when generating the cost volumes. R-MVSNet [8] uses 2D convolutional GRUs to perform sequential cost regularization. It greatly reduces memory consumption since 3D volumes do not need to be held entirely in memory at once.

3 Bi-directional Recurrent MVSNet

In this section, we describe the detailed architecture of our network. The entire design of our proposed network strongly borrows the ideas from R-MVSNet [8].

3.1 Network Architecture

In our proposed method, convolutional GRUs are applied bi-directionally in the depth direction to the cost volumes on the parallel planes divided on the reference camera frustum. The cost volumes have been regularized by aggregating the bi-directional information in each plane by a 2D convolutional layer. The network architecture is shown in Fig. 1.

3.2 Convolutional GRU

For sequential regularization in the proposed method, we use the convolutional GRU, which is the GRU [1] extended to 2D CNN. Let $\mathbf{C}(t)$ be the t-th cost map that divides the cost volume in the depth direction. The output $\mathbf{C}_r(t)$ of the convolutional GRU is formulated as:

$$\mathbf{C}_r(t) = (1 - \mathbf{U}(t)) \otimes \mathbf{C}_r(t-1) + \mathbf{U}(t) \otimes \mathbf{C}_u(t) \quad (1)$$

where $\mathbf{U}(t)$ is the update gate that outputs the update rate from the previous step, $\mathbf{C}_r(t-1)$ is the output of the previous step, $\otimes$ is the element-wise multiplication. Also, $\mathbf{C}_u(t)$ is the current step that has been updated and is represented by the following equation with $\mathbf{W}$ and $\mathbf{b}$ as learnable parameters:

$$\mathbf{C}_u(t) = \sigma_c(\mathbf{W}_c * [\mathbf{C}(t), \mathbf{R}(t) \otimes \mathbf{C}_r(t-1)] + \mathbf{b}_c) \quad (2)$$

where σ_c is a nonlinear function, which is the hyperbolic tangent function. $\mathbf{R}(t)$ is a forget gate that outputs the forget rate from the information in the previous step, $[\cdot]$ is the concatenation, and $*$ represents a convolution operation. The forget gate and the update gate are defined as:

$$\mathbf{R}(t) = \sigma_g(\mathbf{W}_r * [\mathbf{C}(t), \mathbf{C}_r(t-1)] + \mathbf{b}_r) \quad (3)$$

$$\mathbf{U}(t) = \sigma_g(\mathbf{W}_u * [\mathbf{C}(t), \mathbf{C}_r(t-1)] + \mathbf{b}_u) \quad (4)$$

where σ_g is a nonlinear function, which is the sigmoid function. The convolutional GRU propagating in the reverse direction is represented by replacing by $t-1$ with $t+1$.

Although the most basic convolutional GRU as described above consists of a single unit, in our proposed method, three layers of convolutional GRUs are applied to enhance the regularization quality. Besides, Group Normalization [5] is applied before each nonlinear function for more effective information propagation in the recurrent process.

3.3 Aggregation of Two-way Output

The two-way output of GRUs is concatenated and aggregated by a 2D convolutional layer. We denote the forward and backward output as $\mathbf{C}_r^f(t)$ and $\mathbf{C}_r^b(t)$, then the aggregated output $\mathbf{C}_r(t)$ is formulated as:

$$\mathbf{C}_r(t) = \mathbf{W}_a * [\mathbf{C}_r^f(t), \mathbf{C}_r^b(t)] + \mathbf{b}_a \quad (5)$$

where $\mathbf{W}_a$ and $\mathbf{b}_a$ are learnable parameters. Also, $\mathbf{C}_r^f$ and $\mathbf{C}_r^b$ share parameters.

Assuming the cost maps $\{\mathbf{C}\}_0^{D-1}$, the receptive fields of $\mathbf{C}_r^f(t)$ and $\mathbf{C}_r^b(t)$ in the depth dimension are in the range $[0, t]$ and $[t, D-1]$, respectively, and after aggregation, the receptive field of $\mathbf{C}_r(t)$ will be in the range $[0, D-1]$.

3.4 Depth Map Fusion

Depth maps inferred by multi-view images are fused into a single point cloud using the camera parameters. To enhance the quality of the reconstruction, the depth maps are filtered by probability maps and checked for geometric consistency before fusion.

The probability map is an image holding maximum values in the depth direction of a probability volume which is calculated by applying softmax to $\mathbf{C}_r$. In the probability map filtering , pixels in the depth map corresponding to each pixel in the probability map smaller than the pre-defined threshold value will be filtered out.

For geometric consistency checking, we employ Dynamic Consistency Checking [6], which dynamically aggregates geometric matching errors for all views, rather than using a pre-defined strategy or parameters.

4 Implementation

4.1 Datasets

We use the DTU dataset [3], which is used to evaluate MVSNet [7] and R-MVSNet [8], for training, validation and evaluation. DTU dataset is for indoor environment and it consists of 124 scenes captured from 49 camera positions under 7 different lighting conditions.

To measure the reconstruction ability in outdoor environments, we also evaluate on the Tanks and Temples

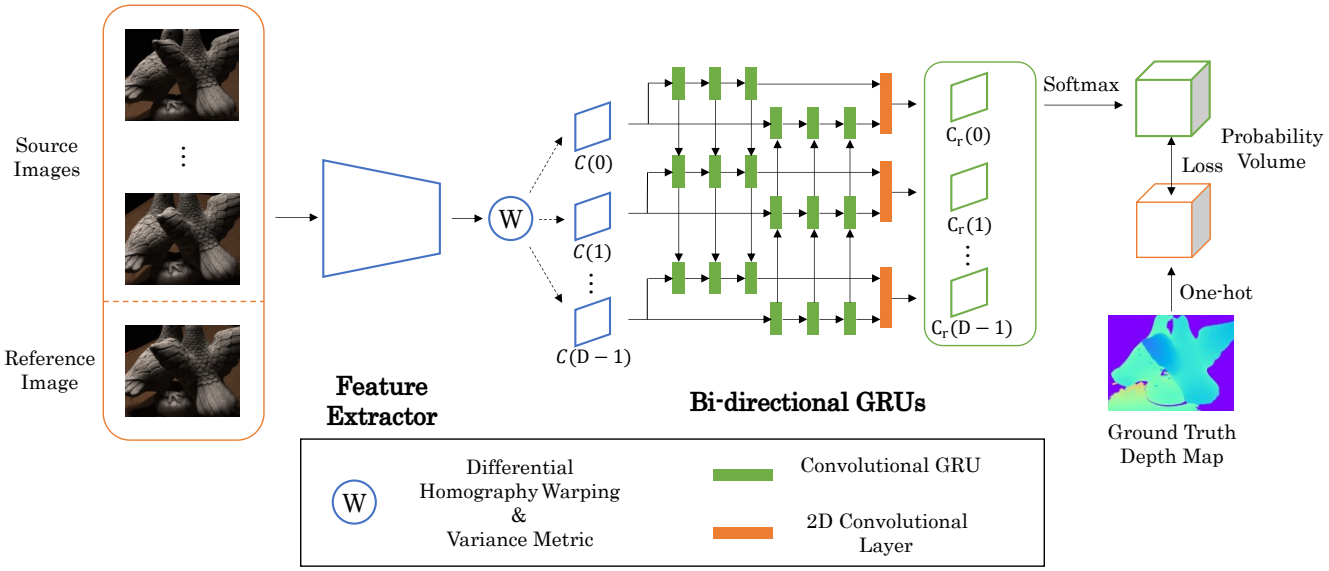


Figure 1. The design of our network. Extracted features from input images are warped into parallel planes on reference camera frustum. We take the variance of corresponding warped features in all views as matching costs. These costs are regularized by bi-directional GRUs in the depth direction to reduce noise and obtain a probability volume. The probability is used for depth map inference and filtering in depth map fusion [6].

intermediate set [4]. This dataset consists of 8 scenes, and 151 to 314 images per scene are used for reconstruction.

4.2 Training

For training, we use the DTU training set [3], where each scan contains 49 images with 7 different lighting conditions. $N = 3$ images are used as input (one reference image and two source images) with image resolution 640×512. The depth sample number is set to $D = 128$, and the cost map is formed in the range of $[425mm, 937mm]$. The learning rate is set to 0.001 and multiplied by 0.9 for every 10,000 steps. We trained our model on an NVIDIA TITAN RTX with a batch size of 1 for 6 epochs. For validation, we use $N = 5$ images as input.

5 Experiments

5.1 DTU Evaluation

In the evaluation on the DTU's evaluation set [3], 22 scenes out of the total 124 scenes are used as the evaluation set. The number of input images N is 5. We fixed the input image resolution to $1,600 \times 1,200$. The inference is performed in the range [425mm, 937mm] with a depth sample size of $D = 256$. The threshold for filtering by probability maps is set to 0.3. The generated depth map size should be 1/4 of the reference image size, due to the feature extractor.

Table 1. Quantitative results on DTU's evaluation set [3]. In this experiments, we implemented R-MVSNet [8] using Dynamic Consistency Checking [6] and denoted it as R-MVSNet+DC. The proposed method outperforms the conventional methods.

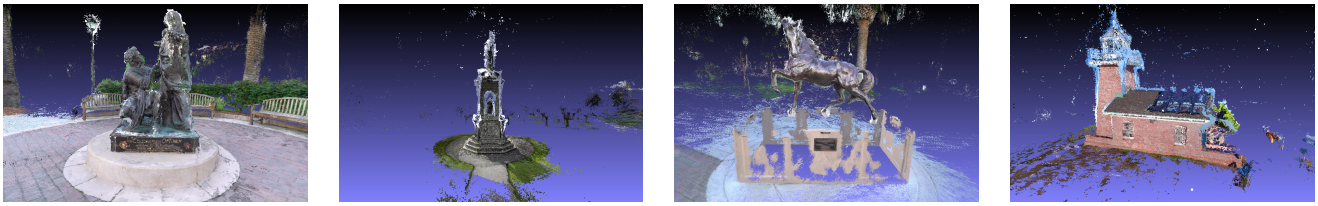
	Distance Metric (mm)		
	Acc.	Comp.	Overall
MVSNet [7]	0.396	0.527	0.462
R-MVSNet [8]	0.385	0.459	0.422
R-MVSNet+DC	0.396	0.382	0.389
Ours	**0.371**	**0.381**	**0.376**

The experimental results on the distance metric are shown in Table 1. It shows an improvement in accuracy (Acc.), completeness (Comp.), and overall scores.

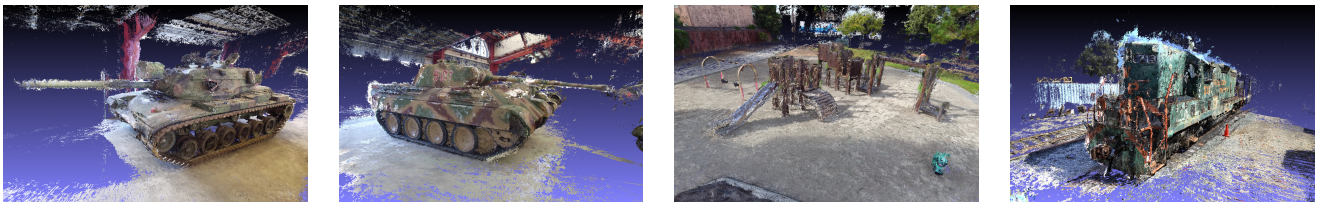
The reconstructed point clouds are shown in Fig. 3. Our method recovered the parts that R-MVSNet [8] had not been able to recover.

5.2 Tanks and Temples Benchmark

On the Tanks and Temples benchmark [4], the input image size is set to $N = 5$ with the image resolution of $1,920 \times 1,080$ or $2,048 \times 1,080$. The depth sample number is set to $D = 256$. The threshold for filtering by probability maps is 0.3. The generated depth map and probability map are bilinearly upsampled to 1/2 of the reference image size and then used for point cloud reconstruction.



(a) Family (b) Francis (c) Horse (d) Lighthouse



(e) M60 (f) Panther (g) Playground (h) Train

Figure 2. Reconstructed point clouds on the Tanks and Temples *intermediate* set [4] produced by our method.

Table 2. Quantitative results on the Tanks and Temples *intermediate* set [4]. These are evaluated in percentage metric and higher is better. L.H. and P.G. stand for Lighthouse and Playground, respectively.

	Percentage Metric (%)								
	Mean	Family	Francis	Horse	L.H.	M60	Panther	P.G.	Train
MVSNet [7]	43.48	55.99	28.55	25.07	50.79	53.96	50.86	47.90	34.69
R-MVSNet [8]	48.40	**69.96**	**46.65**	32.59	42.95	51.88	48.80	52.00	42.38
R-MVSNet+DC	48.27	63.18	39.16	36.38	51.55	51.74	49.12	52.23	42.83
Ours	**50.16**	61.57	41.02	**39.60**	**53.82**	**53.98**	**51.24**	**53.35**	**46.70**

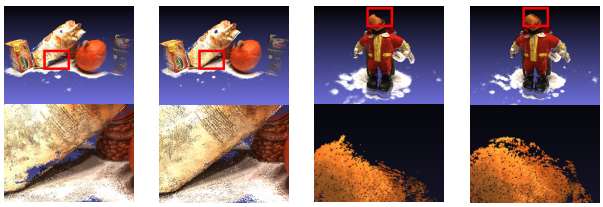


Figure 3. Reconstruction results of point clouds on DTU's evaluation set [3].

The evaluation results of the proposed method are shown in Table 2. Our method improved the percentage metric in all scenes except Family and Francis. The reconstructed point clouds are shown in Fig. 2.

5.3 Running Time and GPU Memory Consumption

We measured the running time and GPU memory consumption of the proposed method and our implemented R-MVSNet [8] during depth inference. The results are shown in Table 3. Both the running time and GPU memory consumption of our method are about 1.3 times longer/larger than that of R-MVSNet [8].

Table 3. Comparison of running time and GPU memory consumption. The image resolution is set to $1,600 \times 1,200$ for ours and R-MVSNet [8], $1,600 \times 1,184$ for MVSNet [7].

	Time (s)	GPU Mem. (MB)
MVSNet [7]	1.32	17,389
R-MVSNet [8]	2.09	1,993
Ours	2.79	2,627

6 Conclusion

In this paper, we have presented an architecture for learning-based multi-view stereo. Our proposed method performs recurrent cost regularization in the depth direction using bi-directional GRUs with shared parameters. It aggregates the bi-directional outputs in a convolutional layer to have a wider receptive field in the depth direction than previous methods. In experiments, our method outperformed conventional methods on the DTU and Tanks and Temples benchmarks [3, 4].

Acknowledgement

This work was supported by JSPS KAKENHI Grant Number JP20J13300.

References

[1] K. Cho, et al.: "Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation" *Empirical Methods in Natural Language Processing*, pp.1724–1734, 2014.

[2] X. Gu, et al.: "Cascade Cost Volume for High-Resolution Multi-View Stereo and Stereo Matching" *Computer Vision and Pattern Recognition*, pp.2495–2504, 2020.

[3] R. Jensen, et al.: "Large Scale Multi-view Stereopsis Evaluation" *Computer Vision and Pattern Recognition*, pp.406–413, 2014.

[4] K. Arno, et al.: "Tanks and Temples: Benchmarking Large-Scale Scene Reconstruction" *ACM Transactions on Graphics*, vol.36, no.4, pp.1–13, 2017.

[5] Y. Wu, et al.: "Group Normalization" *European Conference on Computer Vision*, vol.11217, pp.3–19, 2018.

[6] J. Yan, et al.: "Dense Hybrid Recurrent Multi-view Stereo Net with Dynamic Consistency Checking" *European Conference on Computer Vision*, vol.12349, pp.674–689, 2020.

[7] Y. Yao, et al.: "MVSNet: Depth Inference for Unstructured Multi-view Stereo" *European Conference on Computer Vision*, vol.11212, pp.767–783, 2018.

[8] Y. Yao, et al.: "Recurrent MVSNet for High-resolution Multi-view Stereo Depth Inference" *Computer Vision and Pattern Recognition*, pp.5525–5534, 2019.

[9] M. Ji, et al.: "SurfaceNet: An End-to-end 3D Neural Network for Multiview Stereopsis" *International Conference on Computer Vision*, pp.2307-2315, 2017.

[10] R. Chen, et al.: "Point-Based Multi-View Stereo Network" *International Conference on Computer Vision*, pp.1538-1547, 2019.