Supplementary Material for R-MVSNet

1. Network Architecture

This section describes the network architecture of R-MVSNet (Table 1). R-MVSNet constructs cost maps at different depths, and recurrently regularizes cost maps through the depth direction. The probability volume need to be explicitly computed during the network training, but for testing, we can sequentially retrieve the regularized cost maps and all layers only require the GPU memory with size linear to the input image resolution.

Output	Layer	Input	Output Size
$\{\mathbf{I}_i\}_{i=1}^N$			N×H×W×3
Image Features Extration			
2D_0	ConvBR,K=3x3,S=1,F=8	$\mathbf{I}_i$	H×W×8
2D_1	ConvBR,K=3x3,S=1,F=8	2D_0	H×W× 8
2D_2	ConvBR,K=5x5,S=2,F=16	2D_1	½H×½W×16
2D_3	ConvBR,K=3x3,S=1,F=16	2D_2	½H×½W×16
2D_4	ConvBR,K=3x3,S=1,F=16	2D_3	½H×½W×16
2D_5	ConvBR,K=5x5,S=2,F=32	2D_4	¼H×¼W×32
2D_6	ConvBR,K=3x3,S=1,F=32	2D_5	¼H×¼W×32
$\mathbf{F}_i$	Conv,K=3x3,S=1,F=32	2D_6	¼H×¼W×32
Differentiable Homography Warping			
$\{\mathbf{F}_i, \mathbf{H}_i(d)\}_{i=1}^N$	DH-Warping	$\{\mathbf{V}_i(d)\}_{i=1}^N$	¼H×¼W×32
Cost Map Construction			
$\{\mathbf{V}_i(d)\}_{i=1}^N$	Variance Cost Metric	$\mathbf{C}_0(d)$	¼H×¼W×32
GRU Regularization			
$\mathbf{C}(d)$	Conv,K=3x3,S=1,F=16	$\mathbf{C}_0(d)$	¼H×¼W×16
$\mathbf{C}_0(d)$&$\mathbf{C}_1(d-1)$	GRU, K=3x3, F=16	$\mathbf{C}_1(d)$	¼H×¼W×16
$\mathbf{C}_1(d)$&$\mathbf{C}_2(d-1)$	GRU, K=3x3, F=4	$\mathbf{C}_2(d)$	¼H×¼W×4
$\mathbf{C}_2(d)$&$\mathbf{C}_r(d-1)$	GRU, K=3x3, F=1	$\mathbf{C}_r(d)$	¼H×¼W×1
Probability Volume Construction			
$\{\mathbf{C}_r(d)\}_{d=1}^D$	Softmax	$\{\mathbf{P}_r(d)\}_{d=1}^N$	¼H×¼W×D

Table 1: R-MVSNet architecture. We denote the 2D convolution as Conv and use BR to abbreviate the batch normalization and the Relu. K is the kernel size, S the kernel stride and F the output channel number. N, H, W, D denote input view number, image width, height and depth sample number respectively

2. Depth Sample Number

Given the depth range $[d_{min}, d_{max}]$, we sample depth values using the inverse depth setting:

$$d(i) = \big((\frac{1}{d_{min}} - \frac{1}{d_{max}})\frac{i}{D-1} + \frac{1}{d_{max}}\big)^{-1}, i \in [1, D] \tag{1}$$

where i is the index of the depth sampling and D is the depth sample number. To determine the sample number D, we assume that the spatial image resolution should be the same as the temporal depth resolution. Supposing $\mathbf{X}_1$ and $\mathbf{X}_2$ are two 3D points by projecting the reference image center $(\frac{W}{2}, \frac{H}{2})$ and its neighboring pixel $(\frac{W}{2}+1, \frac{H}{2})$ to the space at depth d_{min}, the spatial image resolution at depth d_{min} is defined as $\rho = ||\mathbf{X}_2 - \mathbf{X}_1||_2$. Meanwhile, we define the temporal depth resolution at depth d_{min} as $d(2) - d(1)$. Considering Equation 1, the depth sample number is calculated as:

$$D = (\frac{1}{d_{min}} - \frac{1}{d_{max}})/(\frac{1}{d_{min}} - \frac{1}{d_{min}+\rho}). \tag{2}$$

3. Variational Depth Map Refinement

We derive the iterative minimization procedure for Equation 8 in the main paper. Focusing on one pixel $\mathbf{p}_1$ in the reference image, we denote its corresponding 3D point in the space as $\mathbf{X} = \mathbf{\Pi}_1^{-1}(\mathbf{p}_1) \cdot d_1 + \mathbf{c}_1$, where $\mathbf{\Pi}_1$, $\mathbf{c}_1$ and d_1 are the projection matrix, camera center of the reference camera and the depth of pixel $\mathbf{p_1}$. The projection of $\mathbf{X}$ in the source image is $\mathbf{p}_i = \mathbf{\Pi}_i(\mathbf{X})$. For the photo-consistency term, we assume $\mathcal{C}(\mathbf{I}_1(\mathbf{p_1}), \mathbf{I}_{i\to1}(\mathbf{p}_1)) = \mathcal{C}(\mathbf{I}_{1\to i}(\mathbf{p}_i), \mathbf{I}_i(\mathbf{p}_i))$ and abbreviate it as $\mathcal{C}_{1\to i}(\mathbf{p}_i)$. The image reprojection error will be changed as $\mathbf{D}_1$ deforms, and we take the derivative of the photo-consistency term w.r.t. to depth d_1:

$$\begin{aligned}
\frac{\partial E^i_{photo}(\mathbf{p}_1)}{\partial d_1} &= \frac{\partial \mathcal{C}_{1\to i}(\mathbf{p}_i)}{\partial d_1} \\
&= \frac{\partial \mathcal{C}_{1\to i}(\mathbf{\Pi}_i(\mathbf{\Pi}_1^{-1}(\mathbf{p}_1)\cdot d_1 + \mathbf{c}_1))}{\partial d_1} \\
&= \frac{\partial \mathcal{C}_{1\to i}(\mathbf{p}_i)}{\partial \mathbf{p}_i} \cdot \frac{\partial \mathbf{p}_i}{\partial \mathbf{X}} \cdot \frac{\partial \mathbf{X}}{\partial d_1} \\
&= \frac{\partial \mathcal{C}_{1\to i}(\mathbf{p}_i)}{\partial \mathbf{p}_i} \cdot \mathbf{J}_i \cdot \mathbf{\Pi}_1^{-1}(\mathbf{p}_1)
\end{aligned} \tag{3}$$

where $\mathbf{J}_i$ is the Jacobian of the projection matrix $\mathbf{\Pi}_i$. $\frac{\partial \mathcal{C}_{1\to i}(\mathbf{p}_i)}{\partial \mathbf{p}_i}$ is the derivative of the photo-metric measurement w.r.t. the pixel coordinate. For computing the derivatives of NCC and ZNCC, we refer readers to [3] for detailed implementations. Also, considering $d_1 = \mathbf{D}_1(\mathbf{p_1})$, the derivative of the smoothness term $\mathcal{S}(\mathbf{p}, \mathbf{p}') = w(\mathbf{p}_1, \mathbf{p}'_1)(\mathbf{D}_1(\mathbf{p}) -$

$\mathbf{D}_1(\mathbf{p}'))^2$ can be derived as:

$$\begin{aligned}
&\frac{\partial E^i_{smooth}(\mathbf{p}_1)}{\partial d_1} \\
&= \sum_{\mathbf{p}'_1 \in \mathcal{N}(\mathbf{p}_1)} w(\mathbf{p}_1, \mathbf{p}'_1) \frac{\partial (\mathbf{D}_1(\mathbf{p}_1) - \mathbf{D}_1(\mathbf{p}'_1))^2}{\partial d_1} \\
&= \sum_{\mathbf{p}'_1 \in \mathcal{N}(\mathbf{p}_1)} 2w(\mathbf{p}_1, \mathbf{p}'_1)(\mathbf{D}_1(\mathbf{p}_1) - \mathbf{D}_1(\mathbf{p}'_1))
\end{aligned} \tag{4}$$

where $w(\mathbf{p}_1, \mathbf{p}'_1) = \exp(-\frac{(\mathbf{I}_1(\mathbf{p}_1) - \mathbf{I}_1(\mathbf{p}'_1))^2}{10})$ is the bilateral smoothness weighting.

We iteratively minimize the total image reprojection error E by gradient descent with a descending step size of $\lambda(t) = 0.9 \cdot \lambda(t-1)$ and $\lambda(0) = 10$. The reference depth map $\mathbf{D}_1$ and all reprojected images $\{\mathbf{I}_{1\to i}\}_{i=2}^N$ will be updated at each step. The refinement iteration is fixed to 20 for all our experiments.

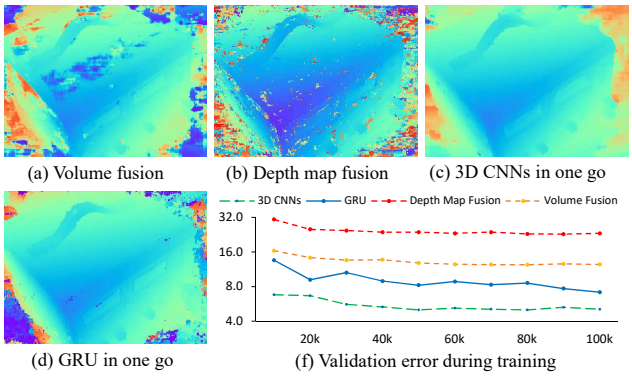
4. Sliding Window 3D CNNs

One concern about R-MVSNet is that whether the proposed GRU regularization could be simply replaced by streaming the 3D CNNs regularization in the depth direction. To address this concern, we conduct two more ablation studies. For DTU dataset, we divide the cost volume $\mathbf{C}$ ($D = 256$) into sub-volumes ($D_{sub} = 64$) along the depth direction. To better regularize the boundary voxels, we set the overlap between two adjacent sub-volumes to $D_{overlap} = 32$, so in this way $\mathbf{C}$ is divided into 7 subsequent sub-volumes $\{\mathbf{C}_i\}_{i=0}^6$. We then sequentially apply 3D CNNs (except for the softmax layer) on $\{\mathbf{C}_i\}_{i=0}^6$ to obtain the regularized sub-volumes. Then, we generate the final depth map by two different fusion strategies:

- **Volume Fusion** First concatenate the regularized sub-volumes (truncated with $D_{trunc} = 16$ to fit the overlap region) in depth direction. Then apply softmax and soft argmin to regress the final depth map.
- **Depth Map Fusion** First regress 7 depth maps and probability maps from the regularized sub-volumes. Then fuse the 7 depth maps into the final depth map by winner-take-all selection on probability maps.

Qualitative and quantitative results are shown in Fig. 2. Both sliding strategies produce errors higher than GRU and 3D CNNs. Also, sliding strategies take $\sim 10s$ to infer depth map ($H \times W \times D = 1600 \times 1184 \times 256$), which is $\sim 2\times$ slower than MVSNet and R-MVSNet.

The sliding window 3D CNNs regularization is a depth-wise divide-and-conquer algorithm and there are two major limitations: 1) One is the discrepancies among sub-volumes, as sub-volumes are not regularized as a whole. 2) The second is the limited size of the sub-volume, which is far less than the actual receptive field size of the multi-scale 3D CNNs ($\sim 256^3$). As a result, such strategies cannot be fully benefit from the powerful 3D CNNs regularization.



(a) Volume fusion (b) Depth map fusion (c) 3D CNNs in one go

(d) GRU in one go (f) Validation error during training

Figure 2: Sliding window 3D CNNs. (a) and (b) are depth map results of the proposed two fusion strategies in A1.

5. Post-processing

We show in Fig. 1 the qualitative point cloud results of DTU *evaluation* set [1] using different post-processing settings. The photo-metric filtering and the geometric filtering are able to remove different kinds of outliers and produce visually clean point clouds. Depth map refinement and depth map fusion have little influence on the qualitative results, however, they are able to reduce the *overall* score for the quantitative evaluation (Table 3 in the main paper).

6. Point Cloud Results

This section presents the point cloud reconstructions of DTU dataset [1], Tanks and Temples benchmark [2] and ETH3D benchmark [4] that have not been shown in the main paper. The point cloud results of the three datasets can be found in Fig. 3, Fig. 4 and Fig. 5 respectively. R-MVSNet is able to produce visually clean and complete point cloud for all reconstructions.

References

[1] H. Aanæs, R. R. Jensen, G. Vogiatzis, E. Tola, and A. B. Dahl. Large-scale data for multiple-view stereopsis. *International Journal of Computer Vision (IJCV)*, 2016. 2, 3, 4

[2] A. Knapitsch, J. Park, Q.-Y. Zhou, and V. Koltun. Tanks and temples: Benchmarking large-scale scene reconstruction. *ACM Transactions on Graphics (TOG)*, 2017. 2, 5

[3] S. Li, S. Y. Siu, T. Fang, and L. Quan. Efficient multi-view surface refinement with adaptive resolution control. *European Conference on Computer Vision (ECCV)*, 2016. 1

[4] 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-

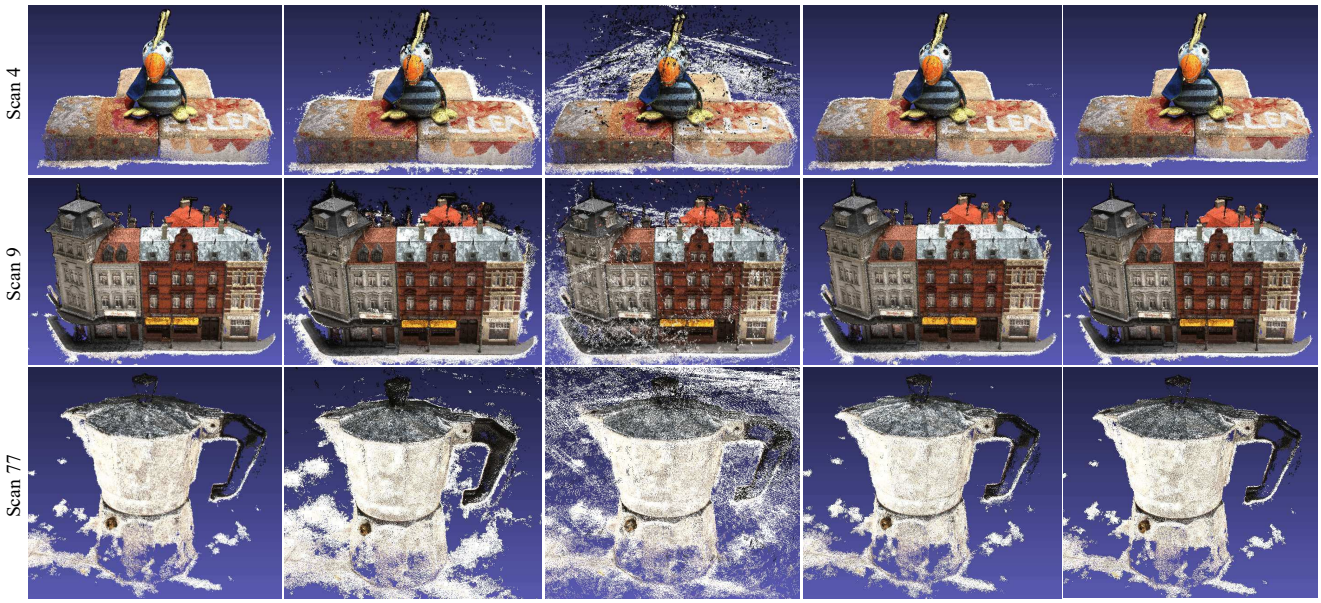


Figure 1: Point cloud reconstructions of DTU dataset [1] with different post-processing settings

camera videos. *Computer Vision and Pattern Recognition (CVPR)*, 2017. 2, 5

Figure 3: Point cloud reconstructions of DTU *evaluation* set [1]

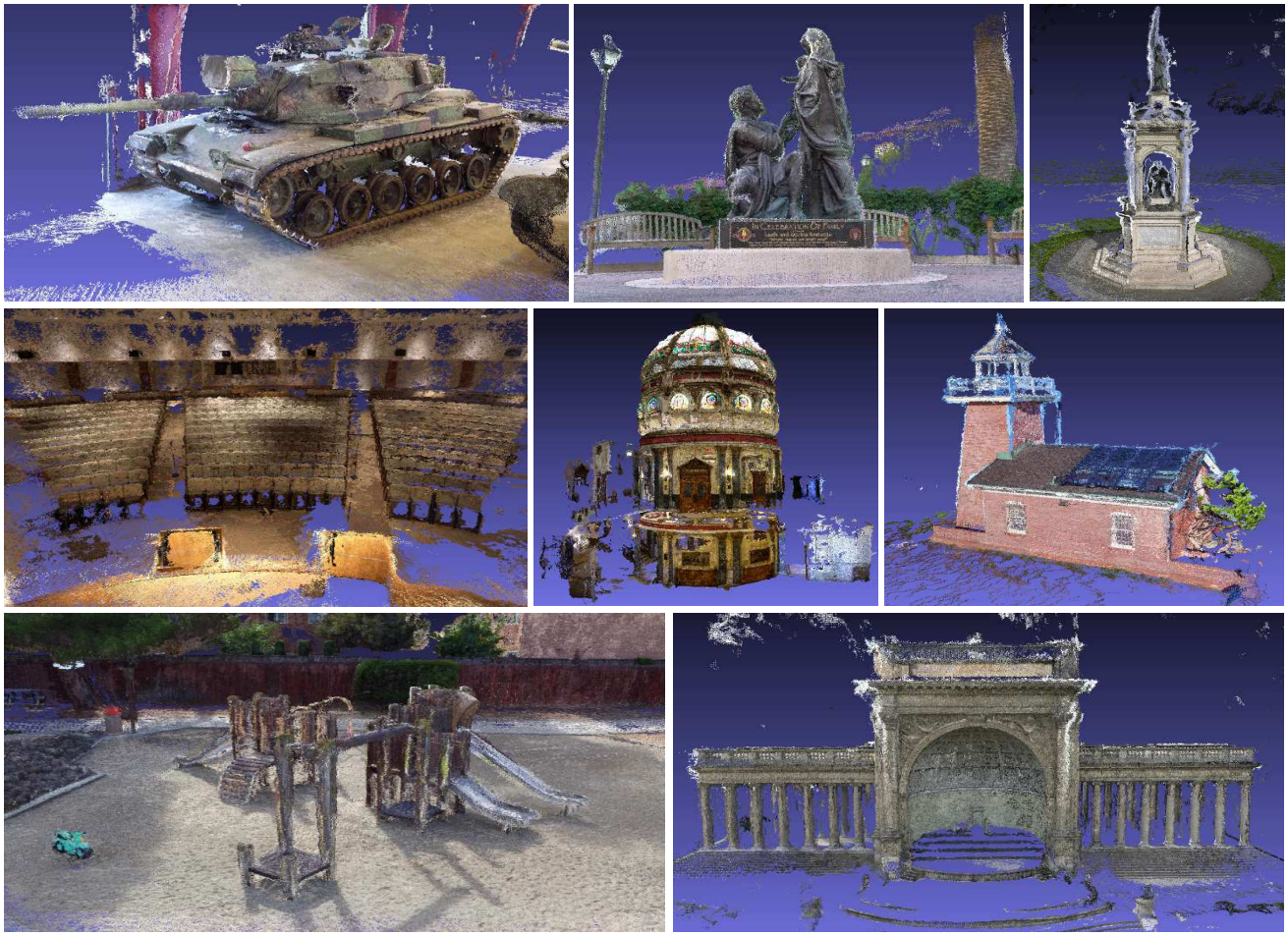


Figure 4: Point cloud reconstructions of Tanks and Temples dataset [2]

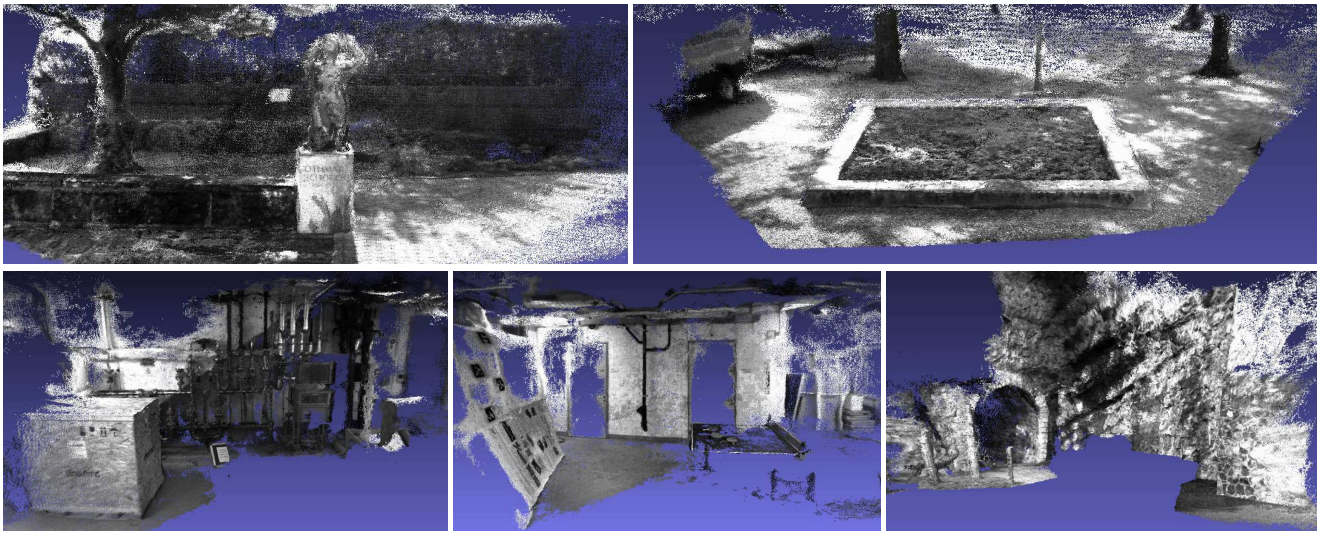


Figure 5: Point cloud reconstructions of ETH3D *low-res* dataset [4]