Efficient Multi-view Stereo by Iterative Dynamic Cost Volume

Shaoqian Wang[♯] Bo Li[♯] Yuchao Dai[*]
School of Electronics and Information, Northwestern Polytechnical University, Xi'an, China

Abstract

In this paper, we propose a novel iterative dynamic cost volume for multi-view stereo. Compared with other works, our cost volume is much lighter, thus could be processed with 2D convolution based GRU. Notably, the every-step output of the GRU could be further used to generate new cost volume. In this way, an iterative GRU-based optimizer is constructed. Furthermore, we present a cascade and hierarchical refinement architecture to utilize the multi-scale information and speed up the convergence. Specifically, a lightweight 3D CNN is utilized to generate the coarsest initial depth map which is essential to launch the GRU and guarantee a fast convergence. Then the depth map is refined by multi-stage GRUs which work on the pyramid feature maps. Extensive experiments on the DTU and Tanks & Temples benchmarks demonstrate that our method could achieve state-of-the-art results in terms of accuracy, speed and memory usage. Code will be released at https://github.com/bdwsq1996/Effi-MVS.

1. Introduction

Multi-view stereo (MVS) aims to reconstruct a dense 3D model based on a series of posed images and corresponding camera parameters. Predicting depth maps and then fusing the depth map into a point cloud model is the most common pipeline of MVS. As a fundamental problem, MVS has been studied for decades in the field of computer vision [5, 8, 10, 12, 16, 23, 24, 27, 28, 34–36]

Recently, we witness a rapid development of deep learning based MVS approaches [5, 10, 12, 23, 28, 34–36]. Generally speaking, the most common basic structure for the MVS task is to use the features of a sequence of images to construct a 3D cost (correlation) volume, then regularize it with 3D CNN, and finally regress the depth. There have been amount of works [3, 4, 17, 31, 32, 35, 36] follow this pipeline and outperform most traditional methods in term of the reconstruction accuracy on MVS benchmarks [1, 13]. However, due to the high GPU memory and processing time requirement of the regularization step, existing methods [31, 35] can only deal with images with low resolution.

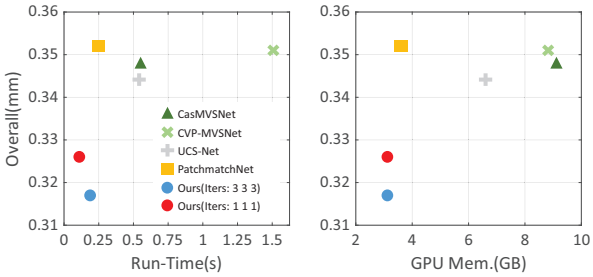


Figure 1. Comparison between our method and SOTA learning-based multi-view stereo methods on the DTU dataset [1]. We report the accuracy in terms of the Overall Error with respect to running time (Left) and GPU memory consumption (Right). The image resolution is 1600×1184. The 'Iters' represent the numbers of iterations at each stage.

Obviously, in addition to improving the reconstruction quality, it is also very important to reduce the running time and GPU memory consumption, which makes the learning-based MVS works adaptable to memory and computational restricted devices. Recently, a couple of works [5, 10, 28, 36, 37] have been proposed to improve the efficiency for MVS. However, it is still very challenging to improve the accuracy and reduce the consumption at the same time. In this work, we aim at improving the computation speed and reducing the memory consumption of high-resolution MVS, and ensuring great reconstruction quality.

The core idea of our method is to construct a lightweight and dynamic cost volume which could be processed in an iterative way. This strategy could bring many benefits. Firstly, it greatly reduces the peak memory usage in the inference phase due to the use of lightweight cost volume. Secondly, the iterative and dynamic processing could guarantee a large search space which is important to the accuracy. At last, our dynamic cost volume is able to converge in a few iterative steps, thus out method is very efficient.

Honestly, our method is partially inspired by the works of [5, 10, 25, 26]. In the work of [5, 10, 25], a cascade adaptive cost volume is presented. In comparison, we further narrow the size of cost volume and extend this multi-

*The first two authors contributed equally. Yuchao Dai is the corresponding author (daiyuchao@gmail.com).

stage strategy to iterative way. Our iterative idea is also inspired by the recent optical estimation work [26]. Compare with [26], we give up the full-size static correlation volume, cause it is not memory friendly. More importantly, different from optical flow estimation, we utilize a lightweight 3D CNN to estimate a coarse depth map as the initialization of GRU, which we find is important to the fast convergence of GRUs on MVS. In addition, due to the 3D CNN is very light, it affects the efficiency slightly.

In conclusion, our contributions can be summarized as:

1) We propose a novel dynamic cost volume, which is very lightweight and could be processed by 2D convolution based GRU iteratively. In this way, we avoid the memory and time consuming problem of large size static cost volume.

2) We present a cascade and hierarchical refinement architecture to utilize the multi-scale information and speed up the convergence. Specifically, with a lightweight 3D CNN, we give a reliable initialization for GRUs, which is critical to fast convergence and final performance.

3) Our method achieves state-of-the-art performance in terms of accuracy, inference speed and GPU memory consumption (*cf*. Fig. 1). As for the accuracy, our method achieves the best results on DTU [1] and advanced sequence of Tanks & Temples dataset [13]. More importantly, with the less memory consumption, our method is 2 times faster than the runner-up.

2. Related Work

2.1. Traditional MVS Methods

Traditional MVS methods can be roughly divided into four categories: voxel-based methods [24, 27], surface-based methods [7, 14], patch-based methods [8, 16], and depth map-based methods [9, 21, 30]. Among those methods, the depth map-based methods have better flexibility and scalability. Gipuma *et al*. [9] proposed a checkerboard-based propagation scheme that is more suitable for parallel computing, and extended the PatchMatch [2] stereo method to multiple views stereo. COLMAP [21] enhances the robustness of the algorithm by jointly estimating pixel-wise view selection, depth map and surface normal. ACMM [30] proposed a multi-scale MVS framework with adaptive checkerboard propagation and multi-hypothesis joint view selection. Although these methods can obtain robust 3D reconstruction results, they suffer from the problems of high computational requirement and poor reconstruction quality.

2.2. Learning based MVS Methods

Recently, there have been amount of deep learning based works proposed for multi-view stereo [5, 10, 12, 28, 34–36]. SurfaceNet [12] is one of the earlier representative works, which builds a voxel volume to aggregate multi-view information and uses 3D CNN to regularize it. Based on differentiable homography [6], Yao *et al*. [35] proposed a widely used deep learning based MVS pipeline. Specifically, MVSNet [35] first extracted feature maps for each view image, then built a cost volume that aggregates the geometric information based on a set of depth hypotheses. In addition, the cost volume was regularized by 3D CNN and the final depth is predicted by regression strategy. However, due to the high memory usage and computation requirement for constructing and regularizing cost volumes, MVSNet can only deal with images with low-resolution. To reduce memory consumption, R-MVSNet [36] utilized recurrent neural network in place of 3D CNN regularization by processing the cost volume along the depth dimension, which reduces the memory requirements but increases the running time. D^2HC-RMVSNet [33] proposed an efficient and effective dense hybrid recurrent multi-view stereo net with dynamic consistency checking. AttMVS [18] proposed a novel attention enhanced matching confidence volume and attention-guided regularization module to improve the matching robustness. Vis-MVSNet [38] explicitly inferred and integrated the pixel-wise occlusion information in the MVS network via the matching uncertainty estimation. AA-RMVSnet [29] proposed an adaptive aggregation module and utilized a hybrid network with recurrent structure for cost volume regularization.

Recently, the efficiency of MVS attracts more and more attention. Fast-MVSNet [37] proposed a novel sparse-to-dense coarse-to-fine framework for fast and accurate depth estimation in MVS. Typically, many works utilized a coarse to fine strategy to reduce the memory consumption and improve the accuracy and speed [5,10,34]. CVP-MVSNet [34] and CasMVSNet [10] built a cascade cost volume based on the pyramid feature map. UCS-Net [5] proposed a strategy of using uncertainty estimation to optimize depth hypotheses. It is worth noting that, very recently, PatchmatchNet [28] introduced the traditional PatchMatch algorithm into the deep learning framework and thus greatly reduced the running time and memory consumption. Comparing with PatchmatchNet, our method could achieve much better reconstruction quality with less running time and memory consumption.

3. Method

As illustrated in Fig. 2, our method consists of a multi-scale feature extractor and GRU-based optimizers. More specifically, the GRU-based optimizer includes a dynamic cost volume constructor and a GRU module. Different from many previous works, our dynamic cost volume could aggregate not only the geometric information but also the context and depth information. More importantly, it could be

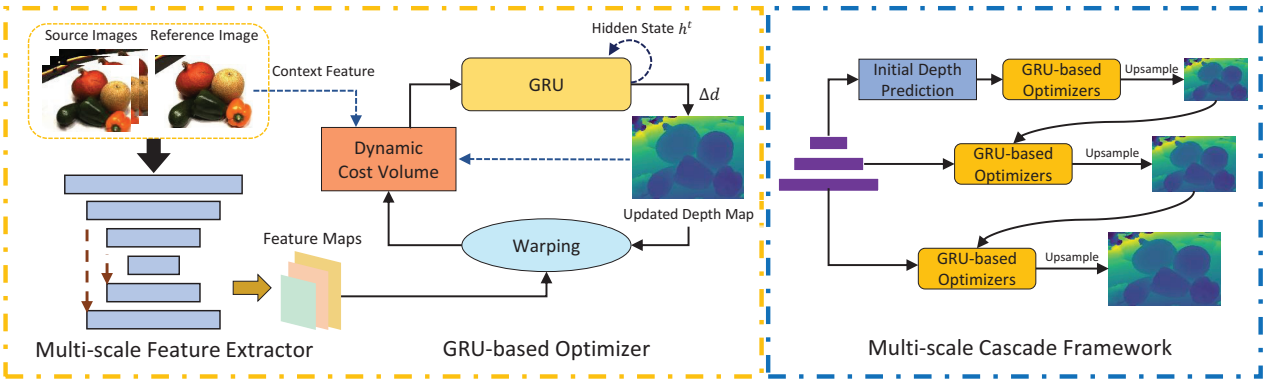


Figure 2. Flowchart of our proposed method. Our method consists of a multi-scale feature extractor and GRU-based optimizers. Specifically, the GRU-based optimizer includes a dynamic cost volume constructor and GRU module. A multi-stage GRUs based cascade architecture is utilized to aggregate multi-scale information and speed up the convergence. Especially, we utilize a lightweight 3D CNN to estimate the initial depth and launch the GRUs.

processed by the GRUs in an iterative way. Specifically, in each updating step, we first construct a proposed dynamic cost volume, and then update the depth map through a GRU. In addition, we present a cascade and hierarchical refinement architecture to utilize the multi-scale information and speed up the convergence. Specifically, a lightweight 3D CNN is utilized to generate the coarsest depth map which could be used as a reliable initialization for the next GRUs. Then the depth map is refined by multi-stage GRUs which work on the pyramid feature maps. It is worth noting that, at each stage k, given the initial depth map $\mathbf{D}_0^k$, the proposed GRU-based optimizer could iteratively update it for several times and output the final depth map. Next, we give more details of our method.

3.1. Multi-scale Feature Extractor

The input images of size $W \times H$ consist of a reference image $\mathbf{I}_0$ and $N-1$ source images $\{\mathbf{I}_i\}_{i=1}^{N-1}$. Similar to [15], we raise a Feature Pyramid Network (FPN) to extract multi-scale features from the N input images. Specifically, the pyramid of feature maps have 3 scale stages $k = 0, 1, 2$. We denote the feature of image I_i at stage k by $\mathbf{F}_i^k$, which is stored at the resolution of $\frac{H}{2^{3-k}} \times \frac{W}{2^{3-k}}$. Likewise, we process the reference image with an additional FPN network to extract the multi-scale context features and initial hidden states for our GRU-based optimizer.

3.2. Dynamic Cost Volume

Cost volume plays critical role in the MVS problem. As presented in Fig. 3, different from the static cost volume in many previous works, we aggregate the geometric feature, depth feature and context feature to construct our dynamic cost volume. The geometric feature, depth feature and context feature are extracted from the local cost volume, depth feature and reference image respectively. More importantly, benefit from the iterative strategy, we can update the depth hypotheses in a narrow inverse depth range to construct the local cost volume in each iteration, which makes our dynamic cost volume is much more lighter compared with the static cost volume.

3.2.1 Local Cost Volume

Similar to previous MVS works [19, 28, 29, 31], given D depth hypotheses $\{d_j | j = 1, ..., D\}$ of the reference view, we construct a local cost volume to represent the correlation between reference and source features. Specifically, for each pixel $\mathbf{p}$ in the reference view, we utilize the differentiable homography to compute the corresponding pixel $\mathbf{p}'_{i,j}$ by warping the source feature $\mathbf{F}_i$ into the j-th depth hypothesis d_j:

$$\mathbf{p}'_{i,j} = \mathbf{K}_i \cdot (\mathbf{R}_{0,i} \cdot (\mathbf{K}_0^{-1} \cdot \mathbf{p} \cdot d_j) + \mathbf{t}_{0,i}) \quad (1)$$

Here $\mathbf{K}_i$ denotes the intrinsic matrix of view i. $\mathbf{R}_{0,i}$ and $\mathbf{t}_{0,i}$ denote the relative rotation and translation parameters between the reference view and source view i. Given $\mathbf{p}'_{i,j}$ and the source feature map $\mathbf{F}_i$, we reconstruct the warped source feature map $\mathbf{F}'_i$ via the differentiable bilinear interpolation. Following MVSNet [35], the cost maps are the variance of $N-1$ warped source feature maps $\{\mathbf{F}'_i\}_{i=1}^{N-1}$.

However, considering that the GRU-based optimizer could iteratively update the depth map, we only sample a few depth hypotheses in a narrow inverse depth range in each iteration. Specifically, for each pixel $\mathbf{p}$ at stage k and iteration t, we uniformly sample $\mathbf{D}_k$ depth hypotheses in the inverse depth range $\mathbf{R}_k$:

$$\mathbf{R}_k = [\frac{1}{\mathbf{D}_{t-1}^k(\mathbf{p})} - 2^{2-k}\mathbf{I}_m, \frac{1}{\mathbf{D}_{t-1}^k(\mathbf{p})} + 2^{2-k}\mathbf{I}_m] \quad (2)$$

Here $\mathbf{I}_m$ represents the minimum depth hypothesis plane interval, which is introduced in Section 4.2 in detail. $\mathbf{D}_{t-1}^k$ is

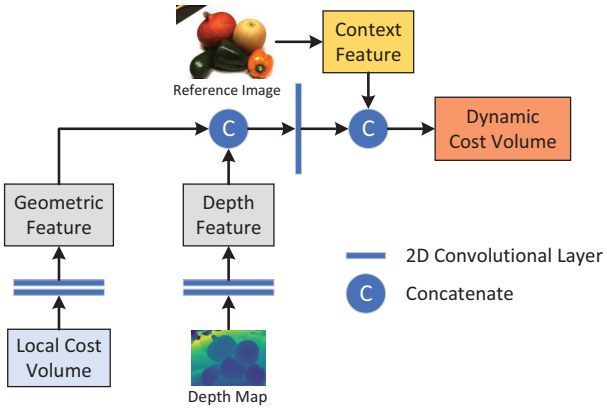


Figure 3. The construction procedure of our dynamic cost volume. We extract the local geometric features from the local cost volume. And fuse it with depth features and context features to form the final dynamic cost volume.

the updated depth map in iteration $t-1$. In order to process the local cost volume with 2D CNN and fuse it with context and depth information, we cancel the depth dimension and concatenate the cost maps along channel dimension. Thus, different from MVSNet [35], the shape of our local cost volume is $\mathbf{C}_L \in R^{W\times H\times(C\times D)}$, where C and D denote the channel and depth dimension respectively.

3.2.2 Features Aggregation

As shown in Fig. 3, in each iteration t, we extract the geometric feature and depth feature from the local cost volume and the depth map $\mathbf{D}^k_{t-1}$ with two lightweight extractors, which consist of two 2D convolution layers. The context feature comes from the multi-scale context feature extractor, and it only need to be extracted once at each stage.

To construct our final dynamic cost volume $\mathbf{C}_D$, we first utilize a 2D convolution layer to process the concatenation of geometric feature and depth feature. Then, the output is concatenated with the context feature to form the dynamic cost volume. All the concatenation operations are performed in the channel dimension.

3.3. Multi-stage GRUs

Our dynamic cost volume is processed by GRUs in an iterative way. In addition, in order to utilize the multi-scale information and speed up the convergence, we construct a multi-stage GRUs architecture. In this way, the dynamic cost volume is constructed on pyramid feature maps and processed by multi-scale GRUs repectively. More specifically, at each stage k, the optimization module will update the depth map for T^k times, and output a set of update amount $\Delta\mathbf{d}^k_t$, where t is from 1 to T^k. In each iteration t, the input depth map $\mathbf{D}^k_{t-1}$ will be updated by

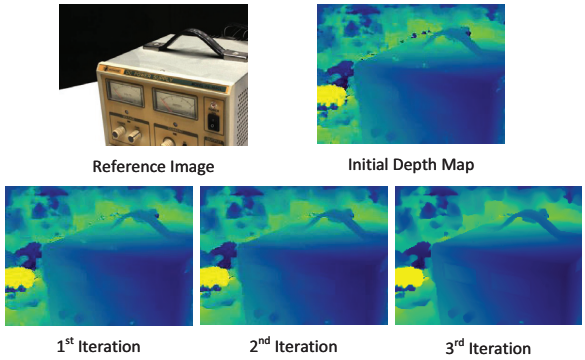


Figure 4. The reference image, initial depth map and the updated depth maps in three iterations at the coarsest stage.

$\mathbf{D}^k_t = \mathbf{D}^k_{t-1} + \Delta\mathbf{d}^k_t$, and then used as the input for the next iteration $t+1$. More importantly, the depth map $\mathbf{D}^k_{T^k}$ in the last iteration at each stage will be upsampled to $\mathbf{D}^k_{T^k+1}$. Then, $\mathbf{D}^k_{T^k+1}$ will be used as the initial depth map $\mathbf{D}^{k+1}_0$ at the next stage.

As shown in Fig. 4, the quality of the depth map could be significantly improved after each iteration. Especially, we could see from the Fig. 4 that the GRU-based optimization module could fill in the holes in the texture-less areas and sharpen the boundary.

3.3.1 Initial Depth Predication

The GRU-based optimization module updates the depth value based on the local spatial information, which makes it sensitive to the initial depth value. Due to the common problems of weak texture regions and similarity regions in MVS, an unreliable initial depth map will make the GRU-based optimization more easier to output wrong depth values. Therefore, we propose an initial depth value prediction module to predict a reliable initial depth value at the coarsest stage.

Inspired by other learning-based MVS methods [19, 28, 29, 31], we predict a probability volume $\mathbf{P}_d$ and corresponding depth map with three modules: cost volume construction, 3D CNN regularization and regression. The architecture of the initial depth predication module is shown in Fig. 5. Following the work of MVSNet [35], we build a tiny cost volume, which consists of sparse depth hypotheses, but includes large enough inverse depth range. Then, we utilize a lightweight 3D CNN to regularize the cost volume and get the probability volume $\mathbf{P}_d$ corresponding to each depth hypothesis d. Finally, we regress the initial depth map $\mathbf{D}^{init}$ with $softargmin$:

$$\mathbf{D}^{init} = \sum_{d=d_{min}}^{d_{max}} d \times \mathbf{P}_d \tag{3}$$

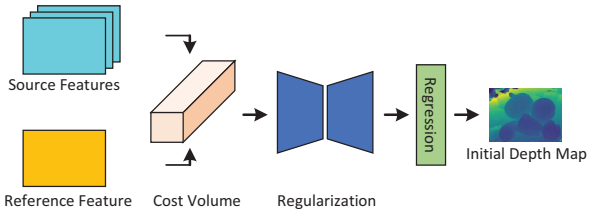


Figure 5. Structure of the initial depth prediction module. We build a cost volume with the coarsest features map, regularize it with a lightweight 3D CNN and regress the initial depth map.

3.3.2 GRU

Inspired by [11,26], we design a GRU module to update the depth map. The details of our GRU module are presented as follows:

$$z^t = \sigma(conv([h^{t-1}, C_D^{t-1}], W_z)) \tag{4}$$

$$r^t = \sigma(conv([h^{t-1}, C_D^{t-1}], W_r)) \tag{5}$$

$$\tilde{h^t} = tanh(conv([r^t \odot h_{t-1}, C_D^{t-1}], W_h)) \tag{6}$$

$$h^t = (1 - r^t) \odot h_{t-1} + z^t \odot \tilde{h^t} \tag{7}$$

Here σ is the $sigmoid$ activation function, W means the parameters for corresponding convolution network, and $conv$ represents a small 2D convolution module, which consist of a 1×5 convolution and a 5×1 convolution.

The inputs of our GRU are the dynamical cost volume $\mathbf{C}_D^{t-1}$ as well as the latent hidden state h^{t-1}. The dynamical cost volume $\mathbf{C}_D^{t-1}$ could be refreshed by the previous depth map $\mathbf{D}_{t-1}^k$. In addition, at each stage, the initial hidden state h^0 is initialized by the context feature network.

Based on the hidden state h^t, we utilize a depth head module to predict the residual depth $\Delta\mathbf{d}_t$. The depth head module contains two convolutional layers and uses the $tanh$ activation function to constrain the range of output values.

After the last iteration T^k at each stage k, we use a mask upsample module [26] to upsample the current depth map($\frac{H}{2^{3-k}} \times \frac{W}{2^{3-k}}$). More specifically, based on the last hidden state h^{T^k}, we utilize two convolutional layers to predict a $\frac{H}{2^{3-k}} \times \frac{W}{2^{3-k}} \times (2 \times 2 \times 9)$ mask, which represents the weights of the neighbors for each pixel. Then the depth map could be upsampled to the resolution of $\frac{H}{2^{2-k}} \times \frac{W}{2^{2-k}}$ by weighted combination based on the predicted mask.

3.4. Loss Function

In training phase, our method could output a couple of depth maps from the the initial depth predication module and multi-stage GRU-based optimization module at different iteration steps. We calculate the $L1$ losses on all output depth maps with the ground truth depth maps of the corresponding resolution. Thus the final loss is the weighted sum of all the losses:

$$L_{total} = L_{init} + \sum_{k=0}^{2} \sum_{i=1}^{T_k+1} \lambda_i^k L_i^k, \tag{8}$$

where L_{init} is the loss of the initial depth map obtained by the initial depth predication module. T_k is the number of optimization iterations at stage k. $\{L_i^k | i = 1...T_k + 1\}$ are the losses of T_k output depth maps and an upsampled depth map at stage k, and λ_i^k is the corresponding weight.

4. Experiments

We evaluated our method on DTU [1] and Tanks & Temples datasets [13]. Extensive experiments are conducted to validate the accuracy and efficiency of our method.

4.1. Datasets

DTU dataset DTU dataset [1] is a large-scale indoor MVS dataset, which contains 128 different scans with 49 views under varying lighting conditions. All the scans are collected under a laboratory environment with same camera trajectory. DTU provides the ground truth depth maps for 79 training scans and the 3D point clouds for 22 evaluation scans. Following the configurations in most MVS work [19, 28, 29, 31], we apply DTU dataset to train and evaluate our network.

Tanks & Temples dataset Tanks & Temples dataset [13] is a large-scale outdoor MVS dataset, which provides a set of video sequences in realistic environments for different scans. It is divided into intermediate and advanced sets, including a total of 14 scenarios. We also evaluate our method on intermediate and advanced sets using the model trained on DTU dataest [1] without fine-tuning.

4.2. Implementation Details

Train During the training on DTU dataset [1], we set the resolution of input images to 640×512 and the number of input images to $N = 5$. For the initial depth predication module, we set the number of depth hypotheses to 48. For the local cost volume, we set the number of depth hypotheses D_k to 4 for all the stage. We define a minimum hypothesis plane interval I_m for inverse depth:

$$I_m = (\frac{1}{d_{min}} - \frac{1}{d_{max}})/Z \tag{9}$$

We set Z to 384, and set the interval of the depth hypotheses at stages 0, 1, 2 to $4I_m, 2I_m, I_m$ (stages 0 is the coarse stage with the resolution of $W \times H = 80 \times 64$). For the optimization module at each stage, we set the iteration number T^k at stages 0, 1, 2 to 3, 3, 3. We train our model with AdamW for 48 epochs under OneCycleLR scheduler with a learning rate of 0.001. We set a batch size of 4 and train our model on 1 NVIDIA GeForce RTX 3090 GPU.

Evaluation We evaluate our proposed method on the DTU evaluation set [1] and both intermediate and advanced

sets of Tanks & Temples dataset [13]. For the evaluation on DTU, we set the number of input images N to 5 and the input images size to 1600 $\times$ 1184. For the evaluation on Tanks & Temples dataset, we use the model trained on DTU without any fine-tuning. We set the number of views N to 7, the input images size to 1920 $\times$ 1056, and the number of depth hypotheses in the initial depth predication module to 96. The camera positions, sparse point cloud, and depth ranges of Tanks & Temples dataset are recovered by the open source SfM software OpenMVG [20].

Filtering and Fusion Similar to other learning-base MVS methods, we filter the output depth map base on the photo-metric and the geometric consistencies. We made some improvements to the filtering algorithm [22], and the specific details are shown in the supplementary material. Meanwhile, we upsampled the probability volume P_d obtained from the initial depth prediction module as a confidence measurement for each pixel, and discard the pixels whose probability of estimated depth is lower than 0.3.

4.3. Benchmark Performance

We compare our method with recent published top-performing learning-based MVS methods in terms of reconstruction quality, running time and GPU memory consumption. As shown in Table. 1, 3, our method achieves the best performance on DTU dataset [1] and advanced sequence of Tanks & Temples dataset [13]. As for the intermediate sequence of Tanks & Temples dataset, we also achieve very competitive results. We further compare our method with recent published learning-based MVS methods [5, 10, 28, 34, 37, 38] who dedicated to improving the efficiency in Table. 2 in the terms of time and memory consumption. We set the same configuration for all methods using the original size images and set the input view number to 5 and 7 on DTU and Tanks & Temples datasets, respectively. As presented in Table. 2, our method is much more efficient in terms of running time and memory consumption. It worth noting that, our fast version (Iters: 1 1 1) is almost 2 times faster than the most closed runner up of PatchmatchNet [28]. More surprisingly, our fast version (Iters: 1 1 1) could still achieve very high reconstruction accuracy which is presented in Table. 1. These experimental results obviously demonstrate that our method could not only improve the computing speed and reducing memory consumption, but also ensuring high-quality reconstruction.

In Fig. 6, We give more quality comparison between our method and some state-of-the-art methods on DTU dataset. From Fig. 6, we can see that our method delivers more accurate boundaries and performed much better on the structure details of the 3D point clouds, compared to CVP-MVSNet [34] and PatchmatchNet [28]. In addition, some 3D point cloud reconstruction results on Tank & Temples dataset are shown in Fig. 7. From Fig. 7, we can see that the reconstruction results of method is of high quality, even for the challenging advanced sequence. At last, more comprehensive quantity results are presented in Table. 3, our method has achieved very competitive results in most evaluation indicators.

Method	Overall(mm)↓	Acc.(mm) ↓	Comp.(mm) ↓
COLMAP [21]	0.532	0.400	0.664
MVSNet [35]	0.462	0.396	0.527
R-MVSNet [36]	0.417	0.385	0.459
CIDER [31]	0.427	0.417	0.437
D^2HC-RMVSNet [33]	0.386	0.395	0.378
AttMVS [18]	0.356	0.383	0.329
AA-RMVSNet [29]	0.357	0.376	0.339
Vis-MVSNet [38]	0.365	0.369	0.361
Fast-MVSNet [37]	0.370	0.336	0.403
CVP-MVSNet [34]	0.351	**0.296**	0.406
CasMVSNet [10]	0.348	0.346	0.351
UCS-Net [5]	0.344	0.338	0.349
PatchmatchNet [28]	0.352	0.427	**0.277**
Ours(Iters:1 1 1)	0.324	0.314	0.334
Ours(Iters:3 3 3)	**0.317**	0.321	0.313

Table 1. The distance metric (lower is better) comparisons on the DTU's evaluation set [1] and mean F-score (higher is better) comparisons on the Tanks & Temples benchmark [13]. The 'Iters' represent the numbers of iterations at each stage.

	DTU		Tanks & Temples	
Method	Time	Mem.	Time	Mem.
Vis-MVSNet [38]	0.61	5.6	0.79	6.3
Fast-MVSNet [37]	0.52	7.0	0.68	7.9
CVP-MVSNet [34]	1.51	8.8	1.79	9.4
CasMVSNet [10]	0.55	9.1	0.72	9.7
UCS-Net [5]	0.54	6.6	0.71	7.5
PatchmatchNet [28]	0.25	3.6	0.35	4.0
Ours(Iters: 3 3 3)	0.19	3.1	0.28	3.6
Ours(Iters: 1 1 1)	**0.11**	**3.1**	**0.19**	**3.6**

Table 2. Comparison of the running time(s) and memory consumption (GB) on DTU [1] and Tanks & Temples benchmark [13] between different methods.

4.4. Ablation Study

In this section, we provide extensive ablation experiments to analyze the impact of the number of iterations, initial depth prediction, components of dynamic cost volume and the number of stages.

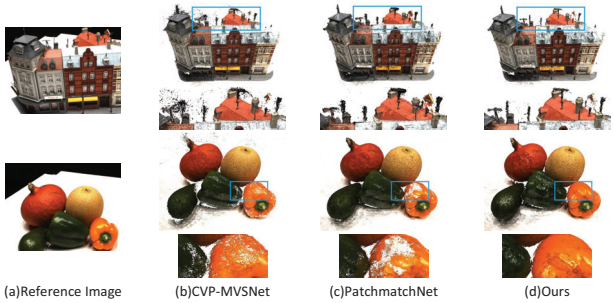
4.4.1 Number of Iterations

Following the same experiment configuration introduced in section 4.2, we conduct ablation experiments with different numbers of iterations. In this way, we would like to verify the effect of number of iterations.

The results are presented in Table. 4. From Table. 4, we could see that the number of iterations is especially impor-

Method	intermediate									advanced						
	Mean	*Fam.*	*Franc.*	*Horse*	*L.H.*	*M60*	*Pan.*	*P.G.*	*Train*	**Mean**	*Audi.*	*Ballr.*	*Courtr.*	*Museum*	*Palace*	*Temple*
COLMAP [21]	42.14	*50.41*	*22.25*	*25.63*	*56.43*	*44.83*	*46.97*	*48.53*	*42.04*	27.24	*16.02*	*25.23*	*34.70*	*41.51*	*18.05*	*27.94*
MVSNet [35]	43.48	*55.99*	*28.55*	*25.07*	*50.79*	*53.96*	*50.86*	*47.90*	*34.69*	-	-	-	-	-	-	-
R-MVSNet [36]	48.40	*69.96*	*46.65*	*32.59*	*42.95*	*51.88*	*48.80*	*52.00*	*42.38*	24.91	*12.55*	*29.09*	*25.06*	*38.68*	*19.14*	*24.96*
CIDER [31]	46.76	*56.79*	*32.39*	*29.89*	*54.67*	*53.46*	*53.51*	*50.48*	*42.85*	23.12	*12.77*	*24.94*	*25.01*	*33.64*	*19.18*	*23.15*
D^2HC-RMVSNet [33]	59.20	*74.69*	*56.04*	*49.42*	*60.08*	*59.81*	*56.61*	*60.04*	*53.92*	-	-	-	-	-	-	-
AttMVS [18]	60.05	*73.90*	***62.58***	*44.08*	***64.88***	*56.08*	*59.39*	***63.42***	***56.06***	31.93	*15.96*	*27.71*	***37.99***	***52.01***	*29.07*	*28.84*
AA-RMVSNet [29]	61.51	***77.77***	*59.53*	*51.53*	*64.02*	***64.05***	***59.47***	*60.85*	*54.90*	33.53	*20.96*	*40.15*	*32.05*	*46.01*	*29.28*	*32.71*
Vis-MVSNet [38]	60.03	*77.40*	*60.23*	*47.07*	*63.44*	*62.21*	*57.28*	*60.54*	*52.07*	33.78	*20.79*	*38.77*	*32.45*	*44.20*	*28.73*	***37.70***
Fast-MVSNet [37]	47.39	*65.18*	*39.59*	*34.98*	*47.81*	*49.16*	*46.20*	*53.27*	*42.91*	-	-	-	-	-	-	-
CVP-MVSNet [34]	54.03	*76.50*	*47.74*	*36.34*	*55.12*	*57.28*	*54.28*	*57.43*	*47.54*	-	-	-	-	-	-	-
CasMVSNet [10]	56.84	*76.37*	*58.45*	*46.26*	*55.81*	*56.11*	*54.06*	*58.18*	*49.51*	31.12	*19.81*	*38.46*	*29.10*	*43.87*	*27.36*	*28.11*
UCS-MVSNet [5]	54.83	*76.09*	*53.16*	*43.03*	*54.00*	*55.60*	*51.49*	*57.38*	*47.89*	-	-	-	-	-	-	-
PatchmatchNet [28]	53.15	*66.99*	*52.64*	*43.24*	*54.87*	*52.87*	*49.54*	*54.21*	*50.81*	32.31	***23.69***	*37.73*	*30.04*	*41.80*	*28.31*	*32.29*
Ours(Iters: 1 1 1)	54.50	*71.12*	*47.51*	*44.38*	*58.19*	*56.59*	*53.45*	*56.72*	*48.03*	32.73	*17.75*	*41.21*	*31.93*	*43.04*	*28.99*	*33.49*
Ours(Iters: 3 3 3)	56.88	*72.21*	*51.02*	***51.78***	*58.63*	*58.71*	*56.21*	*57.07*	*49.38*	**34.39**	*20.22*	***42.39***	*33.73*	*45.08*	***29.81***	*35.09*

Table 3. Detailed quantitative results (higher is better) of different methods on the intermediate set and advanced set of Tanks & Temples benchmark [13]



(a)Reference Image (b)CVP-MVSNet (c)PatchmatchNet (d)Ours

Figure 6. Qualitative comparison of scan 9 and scan 75 in DTU dataset [1]. For each scan, we show the reference image and point cloud results in the top row and the zoomed local areas in the bottom row. Our method delivers more accurate boundaries and provides more denser reconstruction results.

tant for the completeness of the point cloud. It is not surprising, because our dynamic cost volume could aggregate distinctive context information, which could be further utilized by the GRU to fill in the holes. The similar conclusion could also be proven in the Fig. 4, in which the holes are filled with the number of iteration increased. Here, we also would like to argue that our method is rather flexible, i.e., we could adjust the number of iterations in the test phase to balance the reconstruction quality, speed and memory consumption according to actual task requirements.

Iters	Acc.(mm)	Comp.(mm)	Overall(mm)	Time(s)
4,4,4	0.323	**0.311**	0.317	0.23
3,3,3	0.321	0.313	**0.317**	0.19
2,2,2	0.319	0.321	0.320	0.16
1,1,1	**0.314**	0.334	0.324	**0.11**

Table 4. Ablation study of the number of optimization iteration on DTU [1]. The 'Iters' represent the numbers of iterations at each stage.

4.4.2 Initial Depth Prediction

In this experiment, we designed an additional GRU-based optimization module with 4 iterations to replace the initial depth prediction (IDP) module at the coarsest stage. For this optimization module, we set the number of depth hypotheses to 8 and the interval of depth hypotheses to $16\mathbf{I}_m$, The median of the inverse depth range $(d_{min} + d_{max})/2$ is used as the initial input depth map.

Method	Acc.(mm)	Comp.(mm)	Overall(mm)
w/o IDP	0.314	0.374	0.344
Ours	0.321	0.313	0.317

Table 5. Ablation study of the initial depth prediction module(IDP) on DTU's evaluation set [1].

As shown in Table. 5, our method which using the initial depth prediction module obviously performs better in term of the reconstruction quality. Our 3DCNN based initial depth prediction module can obtain a rather reliable depth map, which can effectively avoid the local optimization problem of GRU-based optimization unit. Especially, our method performs much better in the completeness, which further demonstrates the importance of the proposed initial depth prediction module.

4.4.3 Components of Dynamic Cost Volume

In this experiment, we evaluate the beneficial effect of the depth features(DF) and context features(CF) in the construction process of dynamic cost volume on DTU's evaluation set [1]. As shown in Table. 6, the depth features and context features have a great impact on the completeness of the reconstruction results, which can reduce completeness error from 0.368 to 0.313. Without depth features and context features, the dynamic cost volume can only provide geometric feature information extracted from the local cost

Figure 7. Point cloud reconstruction of Tanks & Temples dataset. **Top Row:** the reconstruction results on the intermediate set. **Bottom Row:** the reconstruction results on the advanced set.

volume, which leads to the performance of our method is worse than UCS-Net [5].

Method	Acc.(mm)	Comp.(mm)	Overall(mm)
w/o DF & CF	0.326	0.368	0.347
w/o DF	0.323	0.357	0.340
w/o CF	**0.319**	0.325	0.322
Ours	0.321	**0.313**	**0.317**

Table 6. Ablation study concerning the depth feature(DF) and context feature(CF) in the construction process of dynamic cost volume.

The experimental results also prove that constructing dynamic cost volume by aggregating additional depth feature information and context feature information is the key for our method to achieve the state-of-the-art performance on DTU's evaluation set.

4.4.4 Number of Stages

We further evaluate our method under a total stages number of 2 and 4, and the resolution of the corresponding finest stage is $\frac{H}{4} \times \frac{W}{4}$ and $H \times W$, respectively. We evaluate all the models on the DTU [1] and Tanks & Temples [13] in terms of reconstruction quality, running time and GPU memory consumption. Significantly, we set different ratios of the up sampling module to make all models output full resolution depth maps. As shown in Table. 7, with the increase of the number of stages, the performance has improved significantly on DTU and Tanks & Temples benchmark, but at the same time, the running time and GPU memory consumption have also increased significantly.

		DTU			Tanks & Temples			
Stages	Iters	Overall	Time	Mem.	Inter.	Ad.	Time	Mem.
2	4,4	0.339	0.14	2.7	54.5	33.3	0.23	3.4
3	3,3,3	0.317	0.19	3.1	56.9	34.4	0.28	3.6
4	3,3,2,2	0.314	0.31	4.6	58.8	35.2	0.45	5.2

Table 7. Performance of our method with different stages in terms of the running time(s per view) and memory consumption (GB) on DTU [1] and Tanks & Temples [13] benchmark. The 'Iters' represent the numbers of iterations at each stage from coarse to fine.

5. Conclusion

In this work, we present a novel iterative dynamic cost volume which could be processed by our proposed multi-stage GRUs. Our method is very efficient that could work on very high resolution images. Compared with other learning-based MVS methods, our method achieve the state-of-the-art results in terms of accuracy, speed and memory usage. In addition, our method is very flexible which could get better balance between the accuracy and efficiency through adjusting the iteration numbers in the test phase. In the future, we would like to utilize more powerful feature extractor to improve the performance further.

Acknowledgment

This work was partly supported by the National Natural Science Foundation of China (62001394, 61871325) and the National Key Research and Development Program of China (2018AAA0102803).

References

[1] Henrik Aanæs, Rasmus Ramsbøl Jensen, George Vogiatzis, Engin Tola, and Anders Bjorholm Dahl. Large-scale data for multiple-view stereopsis. *International Journal of Computer Vision*, 120(2):153–168, 2016. 1, 2, 5, 6, 7, 8

[2] Connelly Barnes, Eli Shechtman, Adam Finkelstein, and Dan B Goldman. Patchmatch: A randomized correspondence algorithm for structural image editing. *ACM Trans. Graph.*, 28(3):24, 2009. 2

[3] Rui Chen, Songfang Han, Jing Xu, and Hao Su. Point-based multi-view stereo network. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1538–1547, 2019. 1

[4] Rui Chen, Songfang Han, Jing Xu, and Hao Su. Visibility-aware point-based multi-view stereo network. *IEEE transactions on pattern analysis and machine intelligence*, 43(10):3695–3708, 2020. 1

[5] Shuo Cheng, Zexiang Xu, Shilin Zhu, Zhuwen Li, Li Erran Li, Ravi Ramamoorthi, and Hao Su. Deep stereo using adaptive thin volume representation with uncertainty awareness. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2524–2534, 2020. 1, 2, 6, 7, 8

[6] Robert T Collins. A space-sweep approach to true multi-image matching. In *Proceedings CVPR IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 358–363. IEEE, 1996. 2

[7] Yasutaka Furukawa and Jean Ponce. Carved visual hulls for image-based modeling. In *European Conference on Computer Vision*, pages 564–577. Springer, 2006. 2

[8] Yasutaka Furukawa and Jean Ponce. Accurate, dense, and robust multiview stereopsis. *IEEE transactions on pattern analysis and machine intelligence*, 32(8):1362–1376, 2009. 1, 2

[9] Silvano Galliani, Katrin Lasinger, and Konrad Schindler. Massively parallel multiview stereopsis by surface normal diffusion. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 873–881, 2015. 2

[10] Xiaodong Gu, Zhiwen Fan, Siyu Zhu, Zuozhuo Dai, Feitong Tan, and Ping Tan. Cascade cost volume for high-resolution multi-view stereo and stereo matching. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2495–2504, 2020. 1, 2, 6, 7

[11] Xiaodong Gu, Weihao Yuan, Zuozhuo Dai, Chengzhou Tang, Siyu Zhu, and Ping Tan. Dro: Deep recurrent optimizer for structure-from-motion. *arXiv preprint arXiv:2103.13201*, 2021. 5

[12] Mengqi Ji, Juergen Gall, Haitian Zheng, Yebin Liu, and Lu Fang. Surfacenet: An end-to-end 3d neural network for multiview stereopsis. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2307–2315, 2017. 1, 2

[13] Arno Knapitsch, Jaesik Park, Qian-Yi Zhou, and Vladlen Koltun. Tanks and temples: Benchmarking large-scale scene reconstruction. *ACM Transactions on Graphics (ToG)*, 36(4):1–13, 2017. 1, 2, 5, 6, 7, 8

[14] Zhaoxin Li, Kuanquan Wang, Wangmeng Zuo, Deyu Meng, and Lei Zhang. Detail-preserving and content-aware variational multi-view stereo reconstruction. *IEEE Transactions on Image Processing*, 25(2):864–877, 2015. 2

[15] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2117–2125, 2017. 3

[16] Alex Locher, Michal Perdoch, and Luc Van Gool. Progressive prioritized multi-view stereo. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3244–3252, 2016. 1, 2

[17] Keyang Luo, Tao Guan, Lili Ju, Haipeng Huang, and Yawei Luo. P-mvsnet: Learning patch-wise matching confidence aggregation for multi-view stereo. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 10452–10461, 2019. 1

[18] Keyang Luo, Tao Guan, Lili Ju, Yuesong Wang, Zhuo Chen, and Yawei Luo. Attention-aware multi-view stereo. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1590–1599, 2020. 2, 6, 7

[19] Xinjun Ma, Yue Gong, Qirui Wang, Jingwei Huang, Lei Chen, and Fan Yu. Epp-mvsnet: Epipolar-assembling based depth prediction for multi-view stereo. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5732–5740, 2021. 3, 4, 5

[20] Pierre Moulon, Pascal Monasse, Romuald Perrot, and Renaud Marlet. Openmvg: Open multiple view geometry. In *International Workshop on Reproducible Research in Pattern Recognition*, pages 60–74. Springer, 2016. 6

[21] Johannes L Schonberger and Jan-Michael Frahm. Structure-from-motion revisited. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4104–4113, 2016. 2, 6, 7

[22] Johannes L Schönberger, Enliang Zheng, Jan-Michael Frahm, and Marc Pollefeys. Pixelwise view selection for unstructured multi-view stereo. In *European Conference on Computer Vision*, pages 501–518. Springer, 2016. 6

[23] Steven M Seitz, Brian Curless, James Diebel, Daniel Scharstein, and Richard Szeliski. A comparison and evaluation of multi-view stereo reconstruction algorithms. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 519–528, 2006. 1

[24] Sudipta N Sinha, Philippos Mordohai, and Marc Pollefeys. Multi-view stereo via graph cuts on the dual of an adaptive tetrahedral mesh. In *2007 IEEE 11th International Conference on Computer Vision*, pages 1–8. IEEE, 2007. 1, 2

[25] D. Sun, X. Yang, M. Y. Liu, and J. Kautz. Pwc-net: Cnns for optical flow using pyramid, warping, and cost volume. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018. 1

[26] Zachary Teed and Jia Deng. Raft: Recurrent all-pairs field transforms for optical flow. In *European conference on computer vision*, pages 402–419. Springer, 2020. 1, 2, 5

[27] Ali Osman Ulusoy, Michael J Black, and Andreas Geiger. Semantic multi-view stereo: Jointly estimating objects and

voxels. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4531–4540. IEEE, 2017. 1, 2

[28] Fangjinhua Wang, Silvano Galliani, Christoph Vogel, Pablo Speciale, and Marc Pollefeys. Patchmatchnet: Learned multi-view patchmatch stereo. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14194–14203, 2021. 1, 2, 3, 4, 5, 6, 7

[29] Zizhuang Wei, Qingtian Zhu, Chen Min, Yisong Chen, and Guoping Wang. Aa-rmvsnet: Adaptive aggregation recurrent multi-view stereo network. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 6187–6196, 2021. 2, 3, 4, 5, 6, 7

[30] Qingshan Xu and Wenbing Tao. Multi-scale geometric consistency guided multi-view stereo. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5483–5492, 2019. 2

[31] Qingshan Xu and Wenbing Tao. Learning inverse depth regression for multi-view stereo with correlation cost volume. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 12508–12515, 2020. 1, 3, 4, 5, 6, 7

[32] Jianfeng Yan, Zizhuang Wei, Hongwei Yi, Mingyu Ding, Runze Zhang, Yisong Chen, Guoping Wang, and Yu-Wing Tai. Dense hybrid recurrent multi-view stereo net with dynamic consistency checking. In *European Conference on Computer Vision*, pages 674–689. Springer, 2020. 1

[33] J. Yan, Z. Wei, H. Yi, M. Ding, R. Zhang, Y. Chen, G. Wang, and Y. W. Tai. Dense hybrid recurrent multi-view stereo net with dynamic consistency checking. 2020. 2, 6, 7

[34] Jiayu Yang, Wei Mao, Jose M Alvarez, and Miaomiao Liu. Cost volume pyramid based depth inference for multi-view stereo. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4877–4886, 2020. 1, 2, 6, 7

[35] Yao Yao, Zixin Luo, Shiwei Li, Tian Fang, and Long Quan. Mvsnet: Depth inference for unstructured multi-view stereo. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 767–783, 2018. 1, 2, 3, 4, 6, 7

[36] Yao Yao, Zixin Luo, Shiwei Li, Tianwei Shen, Tian Fang, and Long Quan. Recurrent mvsnet for high-resolution multi-view stereo depth inference. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5525–5534, 2019. 1, 2, 6, 7

[37] Zehao Yu and Shenghua Gao. Fast-mvsnet: Sparse-to-dense multi-view stereo with learned propagation and gauss-newton refinement. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1949–1958, 2020. 1, 2, 6, 7

[38] Jingyang Zhang, Yao Yao, Shiwei Li, Zixin Luo, and Tian Fang. Visibility-aware multi-view stereo network. arXiv preprint arXiv:2008.07928, 2020. 2, 6, 7