GC-MVSNet: Multi-View, Multi-Scale, Geometrically-Consistent
Multi-View Stereo

Vibhas K. Vats, Sripad Joshi, David J. Crandall
Indiana University Bloomington
{vkvats, joshisri, djcran}@iu.edu
Md. Alimoor Reza
Drake University
md.reza@drake.edu
Soon-heung Jung
ETRI
zeroone@etri.re.kr

Abstract

Traditional multi-view stereo (MVS) methods rely heavily on photometric and geometric consistency constraints, but newer machine learning-based MVS methods check geometric consistency across multiple source views only as a post-processing step. In this paper, we present a novel approach that explicitly encourages geometric consistency of reference view depth maps across multiple source views at different scales during learning (see Fig. 1). We find that adding this geometric consistency loss significantly accelerates learning by explicitly penalizing geometrically inconsistent pixels, reducing the training iteration requirements to nearly half that of other MVS methods. Our extensive experiments show that our approach achieves a new state-of-the-art on the DTU and BlendedMVS datasets, and competitive results on the Tanks and Temples benchmark. To the best of our knowledge, GC-MVSNet is the first attempt to enforce multi-view, multi-scale geometric consistency during learning.

1. Introduction

Traditional multi-view stereo (MVS) methods such as Gipuma [10], Furu [9], COLMAP [33], and Tola [36] rely on solving for photometric and geometric consistency constraints across multiple views. Recent machine learning-based MVS methods [2, 3, 6, 12, 25, 30, 41, 45, 47–49, 52] use deep networks to extract feature maps and then construct 3D cost volumes to measure similarity between feature maps [12]. Each paper in this stream of machine learning-based approaches has introduced innovations that have significantly improved the quality of depth estimates and point cloud reconstructions, like multi-level feature extraction, attention-based feature matching, similarity-based cost-volume creation, and improved loss formulations. These modern methods use plane sweep volumes to implicitly encode geometric constraints, and perform multi-view geometric consistency checks as a postprocess after inference to filter the depth maps. However, they do not explicitly model multi-view geometric constraints during learning. Instead, learning about multi-view geometric information thus must happen only implicitly.

In this paper, we show, for the first time, that providing the model with explicit multi-view geometric cues using geometric consistency checks across multiple source views during training (see Fig. 1) significantly improves accuracy while significantly lowering the training iteration requirements. We formulate a multi-stage model called GC-MVSNet which learns geometric cues at three scales. At each scale, we introduce a novel multi-view geometric consistency module that performs geometric consistency checks of reference view depth estimates across multiple source views and generates a per-pixel penalty. This penalty
is then combined with per-pixel depth error (estimated using cross-entropy loss at each stage) to generate the final loss.

This formulation of loss function provides abundant geometric cues to accelerate learning of the model. Our extensive experiments show that GC-MVSNet requires nearly half the training iterations needed by other recent models [6, 12, 30, 41, 48]. Our approach also achieves a new state-of-the-art accuracy on DTU [15] and BlendedMVS [50] datasets, and competitive results on Tanks and Temples [20]. To the best of our knowledge, GC-MVSNet is the first attempt to leverage multi-view, multi-scale geometric consistency checks during the training process. We also perform extensive ablation experiments to demonstrate the effectiveness of the proposed approach.

In summary, in this paper:

- We propose a novel multi-view, multi-scale geometric consistency (GC) module during learning that encourages geometric consistency of reference view depth maps across multiple source views.
- We show that this technique reduces the training iteration requirements to nearly half that of other models, by explicitly providing multi-view geometric cues during learning.
- We show that the module is highly general and can be plugged into different MVS pipelines to enhance geometric cues during training.

2. Related Work

The taxonomy proposed by Furukawa and Ponce [9] classifies MVS methods into four primary scene representations: volumetric fields [7, 21, 34, 35], point clouds [2, 22], 3D meshes [8], and depth maps [1, 3, 6, 10, 12, 30, 33, 45, 48, 52]. Depth map-based methods can further be categorized into either traditional techniques based on feature detection and solving for geometric constraints [1, 9, 10, 33], or learning-based methods [3, 6, 12, 30, 45, 48, 52]. The latter have become very popular in the last few years.

Among the learning-based techniques, MVSNet [48] formulates a single-stage MVS pipeline by encoding camera parameters via differential homography to build 3D cost volumes. It requires a huge amount of memory and computation as it uses 3D U-Nets [32] to regularize the cost volume. Subsequent work has taken two main approaches to alleviate this problem: RNNs [41, 44, 46, 49] and coarse-to-fine multi-stage methods [3, 6, 12, 30, 45, 52].

Among the RNN-based methods, R-MVSNet [49] sequentially regularizes the 2D cost maps along the depth direction via gated recurrent units. AA-RMVSNet [41] slices the cost volume along \( D \) depth hypotheses and regularizes the horizontal and vertical components using CNN and ConvLSTMCells, respectively. Xu et al. [44] use RNNs to model global dependencies with non-local depth interactions. Yan et al. [46] couple LSTM and U-Net architectures to regularize multi-scale information.

Coarse-to-fine multi-stage methods [2, 3, 6, 12, 25, 30, 41, 47, 49] have significantly improved the quality of depth estimates and point cloud reconstructions. They initially predict a low-resolution (coarse) depth map and then progressively refine it. For example, inspired by other coarse-to-fine methods [37, 40, 51], CasMVSNet [12] presents a multi-stage formulation of single-stage MVSNet [48], TransMVSNet [6] focuses on feature matching to improve performance over CasMVSNet, UniMVSNet [30] uses unified loss formulation to further improve over CasMVSNet. CVP-MVSNet [47] builds a cost volume pyramid in a coarse-to-fine manner. UCS-Net [12] uses an adaptive thin volume module that uses a smaller number of hypothesis planes to efficiently partition the local depth range within learned small intervals. TransMVSNet [6] uses transformer based [18, 38] feature matching to promote similarity in the extracted features. UniMVSNet [30] unifies the advantages of regression and classification methods by designing a unified focal loss in a multi-stage framework.

While all these methods improve the performance of the multi-stage MVS pipeline by improving specific portions, none of them explicitly models multi-view geometric cues during the learning process. Consequently, during training these models depend on the limited geometric cues available from multiple source views and the cost function formulation. Xu and Tao [45] present a multi-scale geometric consistency-guided MVS method that uses multi-hypothesis joint view selection to leverage structured region information to sample better candidate hypotheses. They hypothesize that the upsampled depth maps of source images can geometrically constrain these estimates, and use reprojection error [33, 53] to indicate this consistency. In this paper, we use forward-backward reprojection with multiple source views to check the geometric consistency of depth estimates and to generate per-pixel penalties for geometrically inconsistent pixels.

3. Methodology

Our goal is to take \( N \) views as input, including a reference image \( I_0 \in \mathbb{R}^{H \times W \times 3} \) and its paired \( N-1 \) source view images \( \{ I_i \}_{i=1}^{N-1} \), along with the corresponding camera parameters \( c_0, \ldots, c_N \), and then to estimate the reference view depth map \( (D_0) \) as the output.

3.1. Network Overview

Fig. 2 shows the architecture of our approach, which we call Geometric Consistency MVSNet (or GC-MVSNet). We use a deformable convolution-based [4] feature pyramid network (FPN) [23] architecture (Sec. 3.4) to extract features from input images in a coarse-to-fine manner in three
stages. At each stage, we build a correlation-based cost volume of shape $H' \times W' \times D'_{h} \times 1$ using feature maps of shape $N \times H' \times W' \times C$, where $H'$, $W'$, and $C$ denote the height, width, and number of channels of a given stage, and $D'_{h}$ is the number of depth hypotheses at the corresponding stage. The cost volume is regularized with a cost regularization network. We use a winner-takes-all strategy to estimate the depth map $D_{0}$ at each stage. At only the coarse stage, we apply feature matching [6] with linear attention [6, 18] to leverage global context information within and between reference and source view features.

We employ the GC module at each stage. The GC module checks the geometric consistency of each pixel in $D_{0}$ across $M$ source views and generates $\xi_{p}$ (Sec. 3.2), a pixel-wise factor that is multiplied with the per-pixel depth error ($\xi_{d}$), calculated using a cross-entropy function. It penalizes each pixel in $D_{0}$ for its inconsistency across $M$ source views to accelerate geometric cues learning during training. TransMVSNet [6] trained with cross-entropy loss (TransMVSNet-B) is our baseline; see Table 7 for different stages of GC-MVSNet.

### 3.2. Multi-View Geometric Consistency Module

GC-MVSNet estimates reference depth maps at three stages with different resolutions. At each stage, the GC module takes $D_{0}$, $M$ source view ground truths $D_{gt}^{1}, \ldots, D_{gt}^{M}$, and their camera parameters $c_{0}, \ldots, c_{M}$ as input (see Alg. 1). The GC module is then initialized with a geometric inconsistency mask sum (or $\text{mask\_sum}$) of zero at each stage. This mask sum accumulates the inconsistency of each pixel across the $M$ source views. For each source view, the GC module performs forward-backward reprojection of $D_{0}$ to generate the penalty and then adds it to the mask sum.

Forward-backward reprojection (FBR), as shown in Alg. 2, is a crucial three-step process. First, we project each pixel $P_{0}$ of $D_{0}$ to its $i^{th}$ neighboring source view using intrinsic ($K_{R}, K_{S}$) and extrinsic ($E_{R}, E_{S}$) camera parameters to obtain corresponding pixel $P_{i}^{gt}$, and denote the corresponding depth map as $D_{i}^{gt}$. Second, we similarly remap $D_{i}^{gt}$ to obtain $D_{i}^{gt}$, and their camera parameters $c_{0}, \ldots, c_{M}$ as input (see Alg. 1). The GC module is then initialized with a geometric inconsistency mask sum (or $\text{mask\_sum}$) of zero at each stage. This mask sum accumulates the inconsistency of each pixel across the $M$ source views. For each source view, the GC module performs forward-backward reprojection of $D_{0}$ to generate the penalty and then adds it to the mask sum.

The GC module is then initialized with a geometric inconsistency mask sum (or $\text{mask\_sum}$) of zero at each stage. This mask sum accumulates the inconsistency of each pixel across the $M$ source views. For each source view, the GC module performs forward-backward reprojection of $D_{0}$ to generate the penalty and then adds it to the mask sum.

Forward-backward reprojection (FBR), as shown in Alg. 2, is a crucial three-step process. First, we project each pixel $P_{0}$ of $D_{0}$ to its $i^{th}$ neighboring source view using intrinsic ($K_{R}, K_{S}$) and extrinsic ($E_{R}, E_{S}$) camera parameters to obtain corresponding pixel $P_{i}^{gt}$, and denote the corresponding depth map as $D_{i}^{gt}$. Second, we similarly remap $D_{i}^{gt}$ to obtain $D_{i}^{gt}$, and their camera parameters $c_{0}, \ldots, c_{M}$ as input (see Alg. 1). The GC module is then initialized with a geometric inconsistency mask sum (or $\text{mask\_sum}$) of zero at each stage. This mask sum accumulates the inconsistency of each pixel across the $M$ source views. For each source view, the GC module performs forward-backward reprojection of $D_{0}$ to generate the penalty and then adds it to the mask sum.

Forward-backward reprojection (FBR), as shown in Alg. 2, is a crucial three-step process. First, we project each pixel $P_{0}$ of $D_{0}$ to its $i^{th}$ neighboring source view using intrinsic ($K_{R}, K_{S}$) and extrinsic ($E_{R}, E_{S}$) camera parameters to obtain corresponding pixel $P_{i}^{gt}$, and denote the corresponding depth map as $D_{i}^{gt}$. Second, we similarly remap $D_{i}^{gt}$ to obtain $D_{i}^{gt}$, and their camera parameters $c_{0}, \ldots, c_{M}$ as input (see Alg. 1). The GC module is then initialized with a geometric inconsistency mask sum (or $\text{mask\_sum}$) of zero at each stage. This mask sum accumulates the inconsistency of each pixel across the $M$ source views. For each source view, the GC module performs forward-backward reprojection of $D_{0}$ to generate the penalty and then adds it to the mask sum.

Forward-backward reprojection (FBR), as shown in Alg. 2, is a crucial three-step process. First, we project each pixel $P_{0}$ of $D_{0}$ to its $i^{th}$ neighboring source view using intrinsic ($K_{R}, K_{S}$) and extrinsic ($E_{R}, E_{S}$) camera parameters to obtain corresponding pixel $P_{i}^{gt}$, and denote the corresponding depth map as $D_{i}^{gt}$. Second, we similarly remap $D_{i}^{gt}$ to obtain $D_{i}^{gt}$, and their camera parameters $c_{0}, \ldots, c_{M}$ as input (see Alg. 1). The GC module is then initialized with a geometric inconsistency mask sum (or $\text{mask\_sum}$) of zero at each stage. This mask sum accumulates the inconsistency of each pixel across the $M$ source views. For each source view, the GC module performs forward-backward reprojection of $D_{0}$ to generate the penalty and then adds it to the mask sum.

Forward-backward reprojection (FBR), as shown in Alg. 2, is a crucial three-step process. First, we project each pixel $P_{0}$ of $D_{0}$ to its $i^{th}$ neighboring source view using intrinsic ($K_{R}, K_{S}$) and extrinsic ($E_{R}, E_{S}$) camera parameters to obtain corresponding pixel $P_{i}^{gt}$, and denote the corresponding depth map as $D_{i}^{gt}$. Second, we similarly remap $D_{i}^{gt}$ to obtain $D_{i}^{gt}$, and their camera parameters $c_{0}, \ldots, c_{M}$ as input (see Alg. 1). The GC module is then initialized with a geometric inconsistency mask sum (or $\text{mask\_sum}$) of zero at each stage. This mask sum accumulates the inconsistency of each pixel across the $M$ source views. For each source view, the GC module performs forward-backward reprojection of $D_{0}$ to generate the penalty and then adds it to the mask sum.

Forward-backward reprojection (FBR), as shown in Alg. 2, is a crucial three-step process. First, we project each pixel $P_{0}$ of $D_{0}$ to its $i^{th}$ neighboring source view using intrinsic ($K_{R}, K_{S}$) and extrinsic ($E_{R}, E_{S}$) camera parameters to obtain corresponding pixel $P_{i}^{gt}$, and denote the corresponding depth map as $D_{i}^{gt}$. Second, we similarly remap $D_{i}^{gt}$ to obtain $D_{i}^{gt}$, and their camera parameters $c_{0}, \ldots, c_{M}$ as input (see Alg. 1). The GC module is then initialized with a geometric inconsistency mask sum (or $\text{mask\_sum}$) of zero at each stage. This mask sum accumulates the inconsistency of each pixel across the $M$ source views. For each source view, the GC module performs forward-backward reprojection of $D_{0}$ to generate the penalty and then adds it to the mask sum.
consistent pixels, $P_{DE\_mask}$ and $RDD_{\_mask}$, by applying thresholds $D_{\_pixel}$ and $D_{\_depth}$, and then take a logical-OR of the two to produce a single mask of inconsistent pixels. These inconsistent pixels are assigned a value 1 and all other pixels, including the consistent and the out-of-scope pixels, are assigned 0 to form a penalty mask. This penalty mask is then added to the mask sum (Alg. 1), which accumulates the penalty mask for each of the $M$ source views to generate a final mask sum with values $\in [0, M]$. Each pixel value indicates the number of inconsistencies of the pixel across the $M$ source views.

From this mask sum, we then generate the inconsistency penalty $\xi_p$ for each pixel. Our initial approach generated $\xi_p$ by dividing the mask sum by $M$ to normalize within the $[0, 1]$. However, we found that using $\xi_p$ itself for element-wise multiplication reduces the contribution of perfectly consistent (zero inconsistency) pixels to zero, preventing further improvement of such pixels. To avoid this, we add 1 so that elements of $\xi_p$ are in $[1, 2]$. A reference view binary mask is applied on initial $\xi_p$ to generate the final $\xi_p$, as shown in Fig. 3.

Occlusion and its impact. Occluded pixels naturally arise in multi-view stereo, since 3D points are often not visible in all views. These occluded pixels have a major impact on geometric constraints, since the reference view pixels whose corresponding 3D points are occluded are penalized as inconsistent. It is thus important to prevent occluded pixels from dominating the geometric consistency losses. While occlusion is sometimes modeled explicitly [17, 29], we found that our approach is naturally robust to occlusion because of the following three considerations. First, we select the closest $M$ source views as defined in MVSNet [48] to minimize the number of occluded pixels in different source views. Then, during FBR, we remap $D_i^R$ to obtain $D_{\_remap}$ and back project it as shown in Alg. 2. Remapping and back projection largely handles extreme cases of occlusion (see Appendix A in Supplemental Material). Finally, we apply reference view binary mask on $\xi_p$, Fig. 3, to restrict penalties only to valid reference view pixels. The combination of these steps helps us deal with occluded pixels and loss explosion.

### Algorithm 2 Forward Backward Reprojection (FBR)

**Inputs:** $D_{0, c}, D_i^R, P_{0,i}^R$

**Output:** $P_{0,i}^{\prime\prime}, D_i^{\prime\prime}$

- $K, E \leftarrow c_0, K_S, E_S \leftarrow c_i^R$
- $D_{(R \rightarrow S)} \leftarrow (R \rightarrow S)$
- $C_i \leftarrow E \cdot E_S \cdot E_R^{-1} \cdot K_R^{-1} \cdot D_0$
- $X_i \leftarrow C_i \cdot (R \rightarrow S)$
- $D_{\_remap} \leftarrow \text{REMAP}(D_i^R, X_i \cdot (R \rightarrow S), Y_i \cdot (R \rightarrow S))$
- $P_{0,i}^{\prime\prime} \leftarrow (K, E \cdot E_S \cdot R \rightarrow S)$
- $P_{0,i}^{\prime\prime} \leftarrow (X_i, Y_i)$

![Figure 3. The final $\xi_p$ is the outcome of elementwise multiplication (⊙) of initial $\xi_p$ and reference view mask. It restricts the penalties within the reference view mask.](image)

### 3.3. Cost Function

Most learning-based MVS methods [12, 47, 54] treat depth estimation as a regression problem and use an $L_1$ loss between prediction and ground truth. Following AA-RMVSNet [41] and UniMVSNet [30], we treat depth estimation as a classification problem and adopt a cross-entropy loss formulation from AA-RMVSNet [41] (see [30] for relative advantages of regression and classification approaches.) The pixelwise penalty $\xi_{\_d}$ is calculated at each stage,

$$\xi_{\_d} = \sum_{p \in \{P_{\_i}\}} \log(P(d_{\_i}(P)))$$

where $\{P_{\_i}\}$ is the subset of pixels with valid ground truth, $P(d_{\_i}(P))$ denotes the estimated probability of the depth hypothesis $d$ at pixel $P$, and $d$ denotes the depth value closest to the ground truth among all hypotheses. We further enhance the one-hot supervision by penalizing each pixel for its inconsistency across different source views. This is implemented using element-wise multiplication (⊙) between $\xi_{\_d}$ and $\xi_p$ at each stage. The mean stage loss, $L_i$, is calculated as,

$$L_{i(\_stage)} = \text{mean}(\xi_{\_p} \odot \xi_{\_d})$$

where $L_{i(\_stage)}$ is the mean stage loss and $\xi_{\_total}$ is the total loss. $\alpha$, $\beta$, and $\gamma$ are the stage-wise weights. This formulation of cost function with pixel-level inconsistency penalty explicitly forces the model to learn to produce multi-view geometrically-consistent depth maps.

### 3.4. Other Modifications

Besides the geometric consistency module, we made two other major modifications to the MVS pipeline. First, while keeping the feature extraction network architecture as FPN, we replaced its regular convolutional layers with deformable layers [4, 55]. Deformable layers are known as FPN, we replaced its regular convolutional layers with deformable layers [4, 55]. Deformable layers are known as FPN, we replaced its regular convolutional layers with deformable layers [4, 55]. This helps extract better features for accelerated learning.

Second, most MVS methods [6, 12, 30, 41, 54] use batch normalization [14] and batch synchronisation during training. As observed in [14], batch normalization provides more consistent and stable training with large batch sizes, but it is inconsistent and has a degrading effect on training with smaller batches. MVS methods are restricted to very...
We evaluate on three datasets with different complexities. DTU [15] is an indoor dataset that contains 128 scenes with 49 or 64 views under 7 lighting conditions and predefined camera trajectories. We follow MVSNet [48] for training, validation, and test splits. BlendedMVS [50] is a large-scale synthetic dataset with 113 indoor and outdoor scenes. It has 106 training scenes and 7 validation scenes. Tanks and Temples [20] is collected from a more complicated and realistic scene, and contains 8 intermediate and 6 advanced scenes. DTU and TnT evaluate using point clouds while BLD evaluates on depth maps.

4.1. Implementation Details

Following the general practice [6], we first train and evaluate our model on DTU. Then, we finetune on BlendedMVS to evaluate on Tanks and Temples. For training on DTU, we set the number of input images \( N = 5 \) and image resolution as \( 512 \times 640 \). The depth hypotheses are sampled from \( 425 \text{mm} \) to \( 935 \text{mm} \) for coarse-to-fine regularization with the number of plane sweeping depth hypotheses for the three stages set to 48, 32, and 8. The corresponding depth interval ratio (DIR) is set as 2.0, 0.8, and 0.4. The model is trained with Adam [19] for 9 epochs with an initial learning rate (\( LR_{DTU} \)) of 0.001, which decays by a factor of 0.5 once after \( 4^{th} \) epoch. For the Geometric Consistency (GC) module, we use \( M=8 \) and set the stage-wise thresholds \( D_{\text{pixel}} \) as 1, 0.5, 0.25 and \( D_{\text{depth}} \) as 0.01, 0.005, 0.0025. We use \( \alpha=\beta=1 \) and \( \gamma = 2 \) for all experiments. We train our model with a batch size of 3 on 8 NVIDIA RTX A6000 GPUs for about 9 hours.

4.2. Experimental Performance

**Evaluation on DTU.** On DTU, we generate depth maps with \( N=5 \) at an input resolution of \( 864 \times 1152 \). We slightly adjust the depth interval ratio (DIR) to 1.6, 0.7, 0.3 to accommodate the resolution change (more on DIR in Appendix C in Supplemental Material) and use the Fusibile algorithm [10] for depth fusion. Table 1 shows quantitative evaluations, where accuracy is the mean absolute distance between the estimated and the ground truth point cloud, completeness measures the opposite (see Appendix F in Supplemental Material), and overall is the average of these metrics, indicating the overall performance of the models. We find that GC-MVSNet achieves the best overall score as well as the best completeness score, when compared to nearly two dozen previous and state-of-the-art techniques. A qualitative evaluation is presented in Fig. 4 on a few sample MVS problems. We find that our model generates denser and more complete point clouds.

<table>
<thead>
<tr>
<th>Method</th>
<th>Acc ↓</th>
<th>Comp ↓</th>
<th>Overall ↓</th>
</tr>
</thead>
<tbody>
<tr>
<td>SurfaceNet [16]</td>
<td>0.450</td>
<td>1.040</td>
<td>0.745</td>
</tr>
<tr>
<td>MVSNet [48]</td>
<td>0.396</td>
<td>0.527</td>
<td>0.462</td>
</tr>
<tr>
<td>P-MVSNet [25]</td>
<td>0.406</td>
<td>0.434</td>
<td>0.420</td>
</tr>
<tr>
<td>R-MVSNet [49]</td>
<td>0.383</td>
<td>0.452</td>
<td>0.417</td>
</tr>
<tr>
<td>Point-MVSNet [2]</td>
<td>0.342</td>
<td>0.411</td>
<td>0.376</td>
</tr>
<tr>
<td>CasMVSNet [12]</td>
<td>0.325</td>
<td>0.385</td>
<td>0.355</td>
</tr>
<tr>
<td>CVP-MVSNet [47]</td>
<td>0.296</td>
<td>0.406</td>
<td>0.351</td>
</tr>
<tr>
<td>UCS-Net [3]</td>
<td>0.338</td>
<td>0.349</td>
<td>0.344</td>
</tr>
<tr>
<td>AA-RMVSNet [41]</td>
<td>0.376</td>
<td>0.339</td>
<td>0.357</td>
</tr>
<tr>
<td>UniMVSNet [30]</td>
<td>0.352</td>
<td>0.278</td>
<td>0.315</td>
</tr>
<tr>
<td>TransMVSNet [6]</td>
<td>0.321</td>
<td>0.289</td>
<td>0.305</td>
</tr>
<tr>
<td>GBi-Net* [28]</td>
<td>0.312</td>
<td>0.293</td>
<td>0.303</td>
</tr>
<tr>
<td>MVSTER [39]</td>
<td>0.350</td>
<td>0.276</td>
<td>0.313</td>
</tr>
<tr>
<td>GC-MVSNet (ours)</td>
<td>0.330</td>
<td>0.260</td>
<td>0.295</td>
</tr>
</tbody>
</table>

Table 1. Quantitative results on DTU evaluation set at 864 × 1152 resolution. Accuracy (Acc), completeness (Comp) and overall are in mm. * means that GBiNet is re-tested with the same post-processing threshold to all scans for fair comparison with other methods. Gray font shows the methods that use scan-specific thresholds for evaluation. Bold and underline represents first and second place, respectively.

**Evaluation on BlendedMVS.** Unlike DTU and Tanks and Temples, evaluation on Blended MVS is usually measured as the quality of depth maps, not the quality of point clouds. We set \( N=5 \), \( M=8 \), image resolution as \( 576 \times 768 \), and number of depth planes \( D=128 \), and finetune for 10 epochs with one-tenth the learning rate we used for DTU (\( LR_{DTU} \)). We follow [5] for evaluation process.

<table>
<thead>
<tr>
<th>Method</th>
<th>EPE ↓</th>
<th>( e_1 ) ↓</th>
<th>( e_3 ) ↓</th>
</tr>
</thead>
<tbody>
<tr>
<td>MVSNet [48]</td>
<td>1.49</td>
<td>21.98</td>
<td>8.32</td>
</tr>
<tr>
<td>CasMVSNet [12]</td>
<td>1.43</td>
<td>19.01</td>
<td>9.77</td>
</tr>
<tr>
<td>Vis-MVSNet [54]</td>
<td>1.47</td>
<td>15.14</td>
<td>5.13</td>
</tr>
<tr>
<td>EPP-MVSNet [26]</td>
<td>1.17</td>
<td>12.66</td>
<td>6.20</td>
</tr>
<tr>
<td>TransMVSNet [6]</td>
<td>0.73</td>
<td>8.32</td>
<td>3.62</td>
</tr>
<tr>
<td>GC-MVSNet (ours)</td>
<td>0.48</td>
<td>0.89</td>
<td>0.97</td>
</tr>
</tbody>
</table>

Table 2. Quantitative comparison on BlendedMVS evaluation set. We follow evaluation steps described in [5]. Bold and underline represents first and second place, respectively.
Figure 4. Visual comparison of reconstructed point clouds of GC-MVSNet with CasMVSNet [12], R-MVSNet [49] and Ground truths. Our method obtains a more complete point cloud. See Appendix H in Supplemental Material for all point clouds.

Table 3. Quantitative results on intermediate and advanced sets of Tanks and Temples [20]. Bold, single-underline, double-underline represent first, second and third places, respectively.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>COLMAP [33]</td>
<td>42.14</td>
<td>50.41</td>
<td>22.25</td>
<td>26.63</td>
<td>56.53</td>
<td>44.83</td>
<td>46.97</td>
<td>48.53</td>
<td>42.04</td>
<td>27.24</td>
<td>16.02</td>
<td>25.23</td>
<td>34.70</td>
<td>41.51</td>
<td>18.05</td>
<td>27.94</td>
</tr>
<tr>
<td>P-MVSNet [25]</td>
<td>55.62</td>
<td>70.04</td>
<td>44.64</td>
<td>40.22</td>
<td>65.20</td>
<td>55.08</td>
<td>55.17</td>
<td>60.37</td>
<td>54.29</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>R-MVSNet [49]</td>
<td>50.55</td>
<td>73.01</td>
<td>54.56</td>
<td>43.42</td>
<td>43.88</td>
<td>46.80</td>
<td>46.69</td>
<td>50.87</td>
<td>45.25</td>
<td>29.55</td>
<td>19.49</td>
<td>31.45</td>
<td>29.99</td>
<td>42.31</td>
<td>22.94</td>
<td>31.10</td>
</tr>
<tr>
<td>Point-MVSNet</td>
<td>50.55</td>
<td>73.01</td>
<td>54.56</td>
<td>43.42</td>
<td>43.88</td>
<td>46.80</td>
<td>46.69</td>
<td>50.87</td>
<td>45.25</td>
<td>29.55</td>
<td>19.49</td>
<td>31.45</td>
<td>29.99</td>
<td>42.31</td>
<td>22.94</td>
<td>31.10</td>
</tr>
<tr>
<td>CasMVSNet [12]</td>
<td>56.84</td>
<td>76.37</td>
<td>58.45</td>
<td>46.26</td>
<td>55.81</td>
<td>56.11</td>
<td>54.06</td>
<td>58.18</td>
<td>49.51</td>
<td>31.12</td>
<td>19.81</td>
<td>38.46</td>
<td>29.10</td>
<td>43.87</td>
<td>27.36</td>
<td>28.11</td>
</tr>
<tr>
<td>CVP-MVSNet [47]</td>
<td>54.03</td>
<td>76.50</td>
<td>47.74</td>
<td>36.34</td>
<td>55.12</td>
<td>57.28</td>
<td>54.28</td>
<td>57.43</td>
<td>47.54</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>UCS-Net [3]</td>
<td>54.83</td>
<td>76.09</td>
<td>53.16</td>
<td>43.03</td>
<td>54.00</td>
<td>55.60</td>
<td>51.49</td>
<td>57.38</td>
<td>47.89</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>AA-RMVSNet [41]</td>
<td>61.51</td>
<td>77.77</td>
<td>59.53</td>
<td>51.53</td>
<td>64.02</td>
<td>64.05</td>
<td>59.47</td>
<td>60.85</td>
<td>54.90</td>
<td>33.53</td>
<td>20.96</td>
<td>40.15</td>
<td>32.05</td>
<td>46.01</td>
<td>29.28</td>
<td>32.71</td>
</tr>
<tr>
<td>UniMVSNet [30]</td>
<td>64.36</td>
<td>81.20</td>
<td>66.43</td>
<td>53.11</td>
<td>63.46</td>
<td>66.09</td>
<td>64.84</td>
<td>62.23</td>
<td>57.52</td>
<td>38.96</td>
<td>28.33</td>
<td>44.36</td>
<td>39.74</td>
<td>52.89</td>
<td>33.80</td>
<td>34.63</td>
</tr>
<tr>
<td>TransMVSNet [6]</td>
<td>65.52</td>
<td>80.92</td>
<td>65.83</td>
<td>56.94</td>
<td>62.54</td>
<td>62.06</td>
<td>60.00</td>
<td>60.20</td>
<td>58.67</td>
<td>37.00</td>
<td>24.84</td>
<td>34.59</td>
<td>34.77</td>
<td>46.49</td>
<td>34.09</td>
<td>36.62</td>
</tr>
<tr>
<td>GBi-Net [28]</td>
<td>61.42</td>
<td>77.97</td>
<td>67.69</td>
<td>51.81</td>
<td>61.25</td>
<td>60.37</td>
<td>55.87</td>
<td>60.67</td>
<td>53.89</td>
<td>37.32</td>
<td>29.77</td>
<td>42.12</td>
<td>36.30</td>
<td>47.69</td>
<td>31.11</td>
<td>36.39</td>
</tr>
<tr>
<td>MVSTER [39]</td>
<td>60.92</td>
<td>80.21</td>
<td>63.51</td>
<td>52.30</td>
<td>61.38</td>
<td>61.47</td>
<td>58.16</td>
<td>58.98</td>
<td>51.38</td>
<td>37.53</td>
<td>26.68</td>
<td>42.14</td>
<td>35.05</td>
<td>49.37</td>
<td>32.16</td>
<td>39.19</td>
</tr>
<tr>
<td>GC-MVSNet(ours)</td>
<td>62.74</td>
<td>80.87</td>
<td>67.13</td>
<td>53.82</td>
<td>61.05</td>
<td>62.60</td>
<td>59.64</td>
<td>58.68</td>
<td>58.48</td>
<td>38.74</td>
<td>25.37</td>
<td>46.50</td>
<td>36.65</td>
<td>49.97</td>
<td>35.81</td>
<td>38.11</td>
</tr>
</tbody>
</table>

Table 3. Quantitative results on intermediate and advanced sets of Tanks and Temples [20]. Bold, single-underline, double-underline represent first, second and third places, respectively.

depth values, and $e_1$ and $e_2$ are the ratio of number of pixels with $L_1$ error larger than 1mm and 3mm, respectively. The significant improvement in depth map estimates corroborates that providing explicit geometric cues during training helps the model learn about multi-view geometric consistency while requiring much less training iteration. See Appendix H of the Supplemental Material for point clouds.

Evaluation on Tanks and Temples. We also test the performance of our model on an outdoor dataset with the Tanks and Temples benchmark. To adapt to this change, we first finetune our model on BlendedMVS and then evaluate on the intermediate and advanced test sets of Tanks and Temples. We use an image resolution of $576 \times 768$, $N=7$, $M=10$, one-tenth the learning rate of DTU ($\frac{1}{10} LR_{DTU}$), and $D=192$ for finetuning. We finetune the model for 12 epochs. The camera parameters and neighboring view selection are used as in R-MVSNet [49] and follow evaluation steps described in CDS-MVSNet [11].

Table 3 presents our quantitative comparison of different methods. GC-MVSNet achieves the third highest spot on the intermediate set and the second highest spot on advanced set evaluation. Fig. 5 shows point clouds visualizing precision and recall comparisons with other MVS methods. See Appendix H in Supplemental Material for point clouds.

4.3. Ablation Study

Having demonstrated the efficacy of our proposed approach relative to the state of the art, we now conduct ablation studies to evaluate the importance of the various components of our model.

Range of $\xi_p$. $\xi_p$ is generated using the mask sum ($mask\_sum$ in Alg. 1). It is the sum of penalties accumulated across the $M$ source views during multi-view geometric consistency check. At this stage, its elements take a discrete value between 0 and $M$. Using mask sum as it is leads to very high penalty per-pixel and consequently, very high loss value. Such a high loss value destabilizes the learning process. We control the magnitude of penalty...
by controlling the range of per-pixel penalty.

We explore two different ranges to control the magnitude of $\xi_p$, $[1, 2]$ and $[1, 3]$. To generate $\xi_p \in [1, 2]$, we divide the mask sum by $M$ and then add 1. To generate $\xi_p \in [1, 3]$, we divide the mask by $M/2$ and then add 1. Table 4 shows the impact of these two ranges of penalty for $M=8$. Since $\xi_p \in [1, 2]$ produces best results, we use it for all other experiments.

**Hyperparameters of GC module.** The GC module has two types of hyperparameters, global and local. In this section, we investigate the effect of these hyperparameters on our results.

The global hyperparameter $M$ is the number of source views across which the geometric consistency is checked, and is the same for all three stages (coarse, intermediate, and refinement stages). For training on DTU, we vary the value of $M$ while keeping $N=5$, i.e. while the MVS method uses only 4 source views to estimate $D_0$, the GC module checks the geometrical consistency of $D_0$ across $M$ source views. It is important to note that the first $N-1$ out of $M$ source views are exactly the same used by GC-MVSNet to estimate $D_0$. We always keep $M \geq N - 1$.

Table 5 presents a quantitative comparison for different values of $M$ and the amount of training iterations required for optimal performance of the model. At $M=N-1=4$, i.e. checking geometric consistency across the same number of source views as used by GC-MVSNet to estimate $D_0$, the model performance significantly improves with sharp decrease in training iteration requirements, as compared to our baseline TransMVSNet-B (Table 7). As we increase the value of $M$ from 4 to 10, the training iteration required by our model further decreases. We find that at $M=8$, which is twice the number of source views used by GC-MVSNet, it achieves its best performance.

The two local hyperparameters, $D_{pixel}$ and $D_{depth}$, are the stage-wise thresholds applied to generate $PDE_{mask}$ and $RDD_{mask}$ in Alg. 1. These values are set to smaller values in the later (finer) stages, providing a stricter penalty

<table>
<thead>
<tr>
<th>$\xi_p$, Range</th>
<th>Acc↓</th>
<th>Comp↓</th>
<th>Overall↓</th>
<th>Opt. Epoch</th>
</tr>
</thead>
<tbody>
<tr>
<td>$[1, 3]$</td>
<td>0.331</td>
<td>0.270</td>
<td>0.3005</td>
<td></td>
</tr>
<tr>
<td>$[1, 2]$</td>
<td>0.330</td>
<td>0.260</td>
<td>0.295</td>
<td></td>
</tr>
</tbody>
</table>

Table 4. Impact of range of $\xi_p$ during training on DTU with $M=8$, $N=5$. Numbers are generated on DTU evaluation set.

Table 5. Quantitative results on DTU evaluation set [15]. $M$ is the number of source views used by the GC module for checking geometric consistency of reference view depth map. Training iteration requirement of the model decreases as $M$ increases.

to geometrically inconsistent pixels at finer resolutions. Table 6 shows the overall performance of GC-MVSNet with a range of different $D_{pixel}$ and $D_{depth}$ thresholds. GC-MVSNet performance remains fairly consistent and it achieves its best performance with $D_{pixel}=1, 0.5, 0.25$ and $D_{depth}=0.01, 0.005, 0.0025$. We use these threshold values for all datasets throughout the paper.

**GC module as a plug-in.** Our Geometric Consistency module is generic can be integrated into many different MVS pipelines. To demonstrate this, we tested it with two very different MVS pipeline, CasMVSNet [12] and TransMVSNet [6]. CasMVSNet treats depth estimation as a regression problem, while TransMVSNet treats it as a classification problem and uses winner-take-all to estimate the final depth map. We purposefully choose different methods to show that the GC module can perform well for both types of formulation. We compare the architectures of GC-MVSNet with TransMVSNet and CasMVSNet in Sec. 5.1.

Table 7 presents the results, showing the impact of adding the GC module as well as the other modifications (deformable convolution-based FPN with group-norm and

---

Figure 5. Precision and recall comparison with other methods [12, 27, 49] for Train on Tanks and Temples benchmark. $\tau$ is the scene-relevant distance threshold. Darker regions indicate larger error encountered with regard to $\tau$. GC-MVSNet shows visual improvements with brighter regions for precision as well as recall metric.
5. Discussion

5.1. Comparison to Related Work

GC-MVSNet vs. TransMVSNet. TransMVSNet [6] uses regular 2D convolution-based FPN (with batch-norm) for feature extraction and employs adaptive receptive field (ARF) modules with deformable layers after feature extraction. It trains using focal loss [24]. GC-MVSNet replaces the combination of regular FPN and ARF modules with deformable FPN (with group-norm and weight-standardization) for feature extraction. It trains with cross-entropy loss and GC module for accelerated learning.

GC-MVSNet vs. CasMVSNet: CasMVSNet [12] proposes a coarse-to-fine regularization technique. It uses regular 2D convolutions-based FPN for feature extraction, generates variance-based cost volume and employ depth regression to estimate $D_0$. The only similarity with our model is that we also use coarse-to-fine regularization.

5.2. Limitations

Like any other MVS method, GC-MVSNet requires hyper-parameter tuning during learning. Hyperparameters like, depth interval ratio, number of stage-wise depth hypothesis, number of initial depth hypothesis, depth interval decay factor, etc. impacts model performance. The GC module hyperparameters, $D_{\text{pixel}}, D_{\text{depth}}$ and $M$, also require tuning to achieve its best performance. Along with the GC module hyperparameters, the quality of ground truth has also a direct impact on its performance as it uses source view ground truth depth maps for multi-view geometric consistency check.

6. Conclusion

In this paper, we present a novel learning-based MVS pipeline, GC-MVSNet, which explicitly models geometric consistency of reference depth maps across multiple source views during training. To the best of our knowledge, this is the first attempt to leverage multi-view multi-scale geometric consistency check during the training process. We show that the GC module is generic and can be plugged into other MVS methods to accelerate their learning as well. We perform extensive experiments and ablation study to show the advantages of GC-MVSNet. We hope that our work will bring some insights about including explicit geometric reasoning during learning.

Acknowledgement: This work was supported by Electronics and Telecommunications Research Institute (ETRI) grant funded by the Korean government. [23ZH1200, The research of the fundamental media-contents technologies for hyper-realistic media space].

Table 6. Overall score on the evaluation set of DTU [15] for different values of $D_{\text{depth}}$ and $D_{\text{pixel}}$. $M$ is fixed at 8. C, I, and R means Coarse, Intermediate and Refine stages.

<table>
<thead>
<tr>
<th>Stages</th>
<th>Methods</th>
<th>Loss</th>
<th>Other</th>
<th>GC</th>
<th>Overall $\downarrow$</th>
<th>Epoch</th>
</tr>
</thead>
<tbody>
<tr>
<td>QC as a plug-in</td>
<td>CasMVSNet</td>
<td>$L_1$</td>
<td>$\times$</td>
<td>$\times$</td>
<td>0.355</td>
<td>16</td>
</tr>
<tr>
<td></td>
<td>TransMVSNet</td>
<td>FL</td>
<td>$\times$</td>
<td>$\times$</td>
<td>0.305</td>
<td>16</td>
</tr>
<tr>
<td></td>
<td>TransMVSNet-B</td>
<td>CE</td>
<td>$\times$</td>
<td>$\times$</td>
<td>0.328</td>
<td>16</td>
</tr>
<tr>
<td></td>
<td>TransMVSNet-B</td>
<td>CE</td>
<td>$\times$</td>
<td>$\checkmark$</td>
<td>0.298</td>
<td>8</td>
</tr>
<tr>
<td></td>
<td>GC-MVSNet</td>
<td>CE</td>
<td>$\checkmark$</td>
<td>$\checkmark$</td>
<td>0.295</td>
<td>9</td>
</tr>
</tbody>
</table>

Table 7. Performance comparison of different MVS methods with different modifications on DTU [15]. $L_1$, FL, CE and Others indicate $L_1$ loss, Focal loss [24], Cross-entropy loss [41] and other modifications from Sec. 3.4, respectively.

weight-standardization) in the original pipeline. To observe the absolute impact of adding these modifications, we do not change anything else in the original pipelines. We observe in the table that applying only the other modification leads to degradation in performance. It indicates that the other modification helps in stabilizing the training process and promoting reproducibility, but has no significant impact on the performance of the model on its own. We also observe a sharp increase in model performance and decrease in training iteration requirements after integrating our GC module into the original pipeline. With GC, training the CasMVSNet pipeline requires only 11 epochs instead of 16 epochs, while TransMVSNet (with GC module) requires only 8 epochs instead of 16 epochs. This corroborates our hypothesis that multi-view geometric consistency significantly reduces training computation because it accelerates learning of geometric cues.

Table 7 also shows different stages of development of GC-MVSNet. TransMVSNet-B uses TransMVSNet pipeline with cross-entropy loss, performs much worse than original TransMVSNet [6] which uses focal loss. With only other modifications, it slightly improves the overall performance of the model but does not impact the training iteration requirements. Only after applying the GC module, independently and with other modifications, we see significant reduction in training iteration requirements as well as a significant improvement in the overall accuracy metric. This clearly shows the significance of multi-view multi-scale geometric consistency check in the GC-MVSNet pipeline.

Acknowledgement: This work was supported by Electronics and Telecommunications Research Institute (ETRI) grant funded by the Korean government. [23ZH1200, The research of the fundamental media-contents technologies for hyper-realistic media space].
References


[27] Xinjun Ma, Yue Gong, Qirui Wang, Jingwei Huang, Lei Chen, and Fan Yu. Epp-mvsnet: Epipolar-assembling based


[52] Anzhu Yu, Wenyue Guo, Bing Liu, Xin Chen, Xin Wang, Xuefeng Cao, and Bingchuan Jiang. Attention aware cost volume pyramid based multi-view stereo network for 3d re-

