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.

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 our 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-
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 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. DHC-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.

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
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 \( D_0 \), 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 \( I_0 \) and \( N - 1 \) source images \( \{I_i\}_{i=1}^{N-1} \). 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 \( F_{i,k} \), which is stored at the resolution of \( \frac{H}{2^k} \times \frac{W}{2^k} \). 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 \( p \) in the reference view, we utilize the differentiable homography to compute the corresponding pixel \( p'_{i,j} \) by warping the source feature \( F_i \) into the \( j \)-th depth hypothesis \( d_j \):

\[
p'_{i,j} = K_i \cdot (R_{0,i} \cdot (K_0^{-1} \cdot p \cdot d_j) + t_{0,i})
\]

Here \( K_i \) denotes the intrinsic matrix of view \( i \). \( R_{0,i} \) and \( t_{0,i} \) denote the relative rotation and translation parameters between the reference view and source view \( i \). Given \( p'_{i,j} \) and the source feature map \( F_i \), we reconstruct the warped source feature map \( F'_i \) via the differentiable bilinear interpolation. Following MVSNets [35], the cost maps are the variance of \( N - 1 \) warped source feature maps \( \{F'_i\}_{i=1}^{N-1} \).

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 \( p \) at stage \( k \) and iteration \( t \), we uniformly sample \( D_k \) depth hypotheses in the inverse depth range \( R_k \):

\[
R_k = [\frac{1}{D^k_{t-1}(p)} - 2^{2-k}I_m, \frac{1}{D^k_{t-1}(p)} + 2^{2-k}I_m]
\]

Here \( I_m \) represents the minimum depth hypothesis plane interval, which is introduced in Section 4.2 in detail. \( D^k_{t-1} \) is...
the updated depth map in iteration \( t - 1 \). In order to process the local cost volume with 3D 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 \( C_L \in \mathbb{R}^{W \times H \times (C \times D)} \), 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 \( D^k_{t-1} \) 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 \( C_D \), 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 \( \Delta d^k_t \), where \( t \) is from 1 to \( T^k \). In each iteration \( t \), the input depth map \( D^k_{t-1} \) will be updated by

\[
D^k_t = D^k_{t-1} + \Delta d^k_t,
\]

and then used as the input for the next iteration \( t + 1 \). More importantly, the depth map \( D^k_{t-1} \) in the last iteration at each stage will be upsampled to \( D^k_{T^k+1} \). Then, \( D^k_{T^k+1} \) will be used as the initial depth map \( D^0_{k+1} \) 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 \( P_d \) 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 \( P_d \) corresponding to each depth hypothesis \( d \). Finally, we regress the initial depth map \( D^{\text{init}} \) with \( \text{softargmin} \):

\[
D^{\text{init}} = \sum_{d=d_{\text{min}}}^{d_{\text{max}}} d \times P_d 
\]

(3)
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:

\[ z^t = \sigma(\text{conv}(h^{t-1}, C^{t-1}_D, W_z)) \]  
\[ r^t = \sigma(\text{conv}(h^{t-1}, C^{t-1}_D, W_r)) \]  
\[ \tilde{h}^t = \tanh(\text{conv}(r^t \odot h_{t-1}, C^{t-1}_D, W_h)) \]  
\[ h^t = (1 - r^t) \odot h_{t-1} + z^t \odot \tilde{h}^t \]

Here \( \sigma \) is the sigmoid activation function, \( W \) means the parameters for corresponding convolution network, and \( \text{conv} \) represents a small 2D convolution module, which consist of a \( 1 \times 5 \) convolution and a \( 5 \times 1 \) convolution.

The inputs of our GRU are the dynamical cost volume \( C^{t-1}_D \) as well as the latent hidden state \( h^{t-1} \). The dynamical cost volume \( C^{t-1}_D \) could be refreshed by the previous depth map \( D^{t-1} \). 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 \( \Delta d_t \). 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 upsampling module [26] to upsample the current depth map \( \frac{H}{2^{k-1}} \times \frac{W}{2^{k-1}} \) more specifically, based on the last hidden state \( h^{T^k} \), we utilize two convolutional layers to predict a \( \frac{H}{2^{k-1}} \times \frac{W}{2^{k-1}} \times (2 \times 2 \times 9) \) mask, which represents the weights of the neighbors for each pixel. Then the depth map could be upsampled to the resolution of \( \frac{H}{2^{k}} \times \frac{W}{2^{k}} \) 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 initial depth predications module and multi-stage GRU-based optimization module at different iteration steps. We calculate the \( L_1 \) 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:

\[ L_{total} = L_{init} + \sum_{k=0}^{T_k} \sum_{i=1}^{T_k+1} \lambda_i^k f_i^k \]

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 \( \lambda_i^k \) 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 dataset [1] without fine-tuning.

4.2. Implementation Details

**Train** During the training on DTU dataset [1], we set the resolution of input images to \( 640 \times 512 \) 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 \( D_h \) to 4 for all the stage. We define a minimum hypothesis plane interval \( I_m \) for inverse depth:

\[ I_m = \frac{1}{d_{min}} - \frac{1}{d_{max}} / Z \]

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 image 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 image size to \( 1920 \times 1056 \), and the number of depth hypotheses in the initial depth prediction 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, 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 method and some state-of-the-art methods on DTU dataset [1] and Tanks & Temples benchmark [13]. The ‘Iters’ represent the numbers of iterations at each stage.

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

<table>
<thead>
<tr>
<th>Method</th>
<th>Mean</th>
<th>Acc. (mm)</th>
<th>Comp. (mm)</th>
<th>Overall (mm)</th>
<th>Time(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>intermediate</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>COLMAP [41]</td>
<td>0.321</td>
<td>0.314</td>
<td>0.317</td>
<td>0.19</td>
<td>0.23</td>
</tr>
<tr>
<td>MVSNet [35]</td>
<td>0.334</td>
<td>0.313</td>
<td>0.317</td>
<td>0.19</td>
<td>0.23</td>
</tr>
<tr>
<td><strong>advanced</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>AA-RMVSNet [29]</td>
<td>0.323</td>
<td>0.314</td>
<td>0.317</td>
<td>0.19</td>
<td>0.23</td>
</tr>
<tr>
<td>Ours (Iters: 1 1 1)</td>
<td>0.321</td>
<td>0.314</td>
<td>0.317</td>
<td>0.19</td>
<td>0.23</td>
</tr>
<tr>
<td>Ours (Iters: 3 3 3)</td>
<td>0.321</td>
<td>0.314</td>
<td>0.317</td>
<td>0.19</td>
<td>0.23</td>
</tr>
</tbody>
</table>

### 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 $16 I_{m}$. The median of the inverse depth range \( \frac{d_{min} + d_{max}}{2} \) is used as the initial input depth map.

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

<table>
<thead>
<tr>
<th>Method</th>
<th>Acc. (mm)</th>
<th>Comp. (mm)</th>
<th>Overall (mm)</th>
</tr>
</thead>
<tbody>
<tr>
<td>w/o IDP</td>
<td>0.314</td>
<td>0.374</td>
<td>0.344</td>
</tr>
<tr>
<td>Ours</td>
<td>0.321</td>
<td>0.313</td>
<td>0.317</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Method</th>
<th>Acc.(mm)</th>
<th>Comp.(mm)</th>
<th>Overall(mm)</th>
</tr>
</thead>
<tbody>
<tr>
<td>w/o DF &amp; CF</td>
<td>0.326</td>
<td>0.368</td>
<td>0.347</td>
</tr>
<tr>
<td>w/o DF</td>
<td>0.323</td>
<td>0.357</td>
<td>0.340</td>
</tr>
<tr>
<td>w/o CF</td>
<td>0.319</td>
<td>0.325</td>
<td>0.322</td>
</tr>
<tr>
<td>Ours</td>
<td>0.321</td>
<td>0.313</td>
<td>0.317</td>
</tr>
</tbody>
</table>

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.

<table>
<thead>
<tr>
<th>Stages</th>
<th>Iters</th>
<th>Overall</th>
<th>Time</th>
<th>Mem.</th>
<th>Inter.</th>
<th>Ad.</th>
<th>Time</th>
<th>Mem.</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>4,4</td>
<td>0.339</td>
<td>0.14</td>
<td>2.7</td>
<td>54.5</td>
<td>33.3</td>
<td>0.23</td>
<td>3.4</td>
</tr>
<tr>
<td>3</td>
<td>3,3,3</td>
<td>0.317</td>
<td>0.19</td>
<td>3.1</td>
<td>56.9</td>
<td>34.4</td>
<td>0.28</td>
<td>3.6</td>
</tr>
<tr>
<td>4</td>
<td>3,3,2,2</td>
<td>0.314</td>
<td>0.31</td>
<td>4.6</td>
<td>58.8</td>
<td>35.2</td>
<td>0.45</td>
<td>5.2</td>
</tr>
</tbody>
</table>

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


