Rethinking Disparity: A Depth Range Free Multi-View Stereo Based on Disparity

Qingsong Yan\textsuperscript{*}.\textsuperscript{1,2}, Qiang Wang\textsuperscript{*}.\textsuperscript{3}, Kaiyong Zhao\textsuperscript{4}, Bo Li\textsuperscript{2}, Xiaowen Chu\textsuperscript{\dagger}.\textsuperscript{2,5}, Fei Deng\textsuperscript{*}.\textsuperscript{1}

\textsuperscript{1} Wuhan University, Wuhan, China
\textsuperscript{2} The Hong Kong University of Science and Technology, Hong Kong SAR, China
\textsuperscript{3} Harbin Institute of Technology (Shenzhen), Shenzhen, China
\textsuperscript{4} XGRIDs, Shenzhen, China
\textsuperscript{5} The Hong Kong University of Science and Technology (Guangzhou), Guangzhou, China

yanqs\_whu@whu.edu.cn, qiang.wang@hit.edu.cn, kyzhao@xgrid.com, bli@cse.ust.hk
xwchu@ust.hk, fdeng@sgg.whu.edu.cn

Abstract

Existing learning-based multi-view stereo (MVS) methods rely on the depth range to build the 3D cost volume and may fail when the range is too large or unreliable. To address this problem, we propose a disparity-based MVS method based on the epipolar disparity flow (E-flow), called DispMVS, which infers the depth information from the pixel movement between two views. The core of DispMVS is to construct a 2D cost volume on the image plane along the epipolar line between each pair (between the reference image and several source images) for pixel matching and fuse uncountable depths triangulated from each pair by multi-view geometry to ensure multi-view consistency. To be robust, DispMVS starts from a randomly initialized depth map and iteratively refines the depth map with the help of the coarse-to-fine strategy. Experiments on DTUMVS and Tanks&Temple datasets show that DispMVS is not sensitive to the depth range and achieves state-of-the-art results with lower GPU memory.

Introduction

Multi-view stereo matching (MVS) is a core technique in 3D reconstruction that has been extensively studied (Furukawa and Ponce 2009; Galliani et al. 2015; Schönberger et al. 2016). Although traditional methods try to introduce additional constraints (Xu and Tao 2019; Romanoni and Matteucci 2019; Xu and Tao 2020a) to deal with textureless regions or repeated textures, they still have difficulty in guaranteeing the generation of high-quality point clouds in many cases.

Recently, learning-based methods have brought a new light to MVS. MVSNet (Yao et al. 2018) shows a fully differentiable pipeline, which firstly uses a convolutional neural network (CNN) to extract features from input images, and then splits the 3D space into several bins covering a certain depth range to build a 3D cost volume by differentiable homograph, and finally relies on a 3D CNN to regress the depth map. Although MVSNet achieves impressive results on several public benchmarks (Aanæs et al. 2016; Knapitsch et al. 2017), it is not efficient and requires a lot of GPU memory.

* These authors contributed equally.
\textsuperscript{\dagger} Corresponding author

Copyright © 2023, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.
tion (Wei et al. 2021), the attention mechanism (Luo et al. 2020), and the Transformer (Zhu et al. 2021) are used to improve the quality of image features. In addition, the visibility information (Xu and Tao 2020b; Zhang et al. 2020) and the epipolar (Ma et al. 2021; Yang et al. 2022) are also used to boost the performance.

A critical issue of those existing methods is the fixed depth range in building the cost volume (Cheng et al. 2020; Mi, Di, and Xu 2022). Usually, the depth range decides the 3D distribution of cost volume that the network attempts to fit, and the size of cost volume is limited to the computational and memory capability, which results in these methods can be easy to over-fit the configured depth range. Figure 1 shows that the quality of point cloud construction by two state-of-the-art methods, GBiNet (Mi, Di, and Xu 2022) and IterMVS (Wang et al. 2022), is dramatically degraded when the depth range is enlarged. The reason is that these methods cannot capture enough matching information with a fixed number of depth bins.

In this paper, we propose a new MVS pipeline, which allows the CNN to focus only on the matching problem between two different views and relies on the multi-view geometry to recover the depth by matching results. The contributions of this paper are as follows.

- Instead of constructing the 3D cost volume, this paper only constructs the 2D cost volume to match pixels between each pair and generates the depth map by triangulation. In other words, DispMVS exploits the multi-view geometry to reduce the burden of networks, which does not rely on the computationally expensive 3D cost volume to estimate the depth.
- We redesign the flow to deal with the multi-view stereo matching without applying stereo rectification for each pair. First, we propose the epipolar disparity flow (E-flow) and reveal the relationship between the E-flow and the depth. Then we extend E-flow from two-view to multi-view and iteratively update the E-flow by a fused depth to maintain multi-view consistency.
- DispMVS achieves the state-of-the-art result on the DTUMVS and Tanks&Temple without the 3D cost volume, demonstrating the effectiveness of combining multi-view geometry with a learning-based approach.

### Related Work

With decades of development of MVS, many traditional and learning methods are proposed. Traditional MVS cannot surpass the limitations of the artificially designed matching pipeline and fail to reconstruct the non-Lambertian regions. On the contrary, learning-based methods can automatically find the most helpful information in a data-driven manner, and the benchmarking results of (Aanæs et al. 2016; Knäpisch et al. 2017) show that the learning-based method can easily outperform traditional methods. Generally, the learning-based methods build a 3D cost volume, and we can categorize them into 3D convolution-based and RNN-based methods according to how they handle the 3D cost volume.

### Traditional MVS

Traditional MVS has three major types: volumetric method, point cloud method, and depth map method. The volumetric method (Seitz and Dyer 1999; Kostrikov, Horbert, and Leibe 2014) splits the space into inside and outside and cuts out the surface. The point cloud method (Lhuillier and Quan 2005; Furukawa and Ponce 2009) reconstructs dense point clouds from sparse point clouds. Although these methods can reconstruct high-quality results, the volumetric and point cloud methods require lots of GPU memory and are hard to parallelize. The depth map method is the most popular, which separately reconstructs the depth map of each view and fuses them to generate the point cloud (Galliani et al. 2015). Shen (Shen 2013) and Colmap (Zheng et al. 2014; Schönberger et al. 2016) extends the PatchMatch (Bleyer, Rhemann, and Rother 2011) to multi-view and simultaneously estimates the normal and the depth. Meanwhile, Gipuma (Galliani et al. 2015) and ACM (Xu and Tao 2019) use a GPU to improve the computational efficiency. The superpixel (Romanoni and Matteucci 2019), the plane (Xu and Tao 2020a), and the mesh (Wang et al. 2020) are used to reduce mismatching in non-Lambertian regions. Although these methods can achieve stable results, they cannot surpass the limitation of hand-craft methods in a challenging environment.

### 3D Convolution Method

Various approaches have been proposed to address the shortcomings of MVSNet (Yao et al. 2018).

The straightforward solution to reduce the GPU memory is building fewer 3D cost volume. Fast-MVSNet (Yu and Gao 2020) only calculates the 3D cost volume on a sparse depth map and propagates the sparse depth map into a dense depth map. CasMVSNet (Gu et al. 2020) and CVP-MVSNet (Yang et al. 2020) uses a coarse-to-fine strategy to deal with the 3D cost volume and reduce computation cost on the high resolution. GBiNet (Mi, Di, and Xu 2022) treats the MVS as a binary search problem and only builds the 3D cost volumes on the side with a high probability of containing the depth.

Various methods are proposed to improve the 3D cost volume quality. P-MVSNet (Luo et al. 2019) uses the isotropic and anisotropic 3D convolution to estimate the depth map. AttMVS (Luo et al. 2020) applies attention mechanism into the 3D cost volume to improve the robustness. UCS-MVSNet (Cheng et al. 2020) uses the uncertainty as a guide to adjust the 3D cost volume. EPP-MVSNet (Ma et al. 2021) proposes an epipolar-assembling module to enhance the 3D cost volume. MVSTR (Zhu et al. 2021) relies on the 3D-geometry transformer (Dosovitskiy et al. 2021) to obtain global context and 3D consistency. Also, considering the occlusion, Vis-MVSNet (Zhang et al. 2020) and PVSNet (Xu and Tao 2020b) introduce the visibility to filter out unreliable regions. Besides, MVSCRF (Xue et al. 2019) uses a conditional random field to ensure the smoothness of the depth map, and Uni-MVSNet (Peng et al. 2022) combines the regression and classification by the unified focal loss.

In addition, unsupervised MVS has also achieved impressive results. Un-MVSNet (Khot et al. 2019) uses photometric as a guide to learning the depth map. MVSNet2 (Dai et al. 2019) uses multi-view depth maps to filter out occlu-
Figure 2: The pipeline of DispMVS. After extracting features from input images, DispMVS first uses a random depth map to initialize the E-flow between each pair and then triangulates a new depth map by the E-flow that is updated through a GRU module. Finally, with several iterations, DispMVS can reconstruct a high-quality depth map.

RNN Method Instead of using the 3D CNN to process the 3D cost volume, the RNN-based method uses the more efficient LSTM or GRU. R-MVSNet (Yao et al. 2019) uses the GRU to regularize the cost volume sequentially. RED-Net (Liu and Ji 2020) utilizes a 2D recurrent encoder-decoder structure to process the cost volume based on the GRU module. D2HC-RMVSNet (Yan et al. 2020) and AA-RMVSNet (Wei et al. 2021) proposes a hybrid architecture that combines the LSTM and the UNet (Ronneberger, Fischer, and Brox 2015). PatchMatchNet (Wang et al. 2021) introduces an end-to-end PatchMatch (Bleyer, Rhemann, and Rother 2011) with adaptive propagation during each iteration and achieves competitive performance with lower GPU memory. Recently, IterMVS (Wang et al. 2022) uses RAFT (Teed and Deng 2020; Lipson, Teed, and Deng 2021) as backbone and iterative updates the depth map based on the GRU module.

Method

In this section, we introduce the details of the proposed method. The pipeline of DispMVS is demonstrated in Figure 2. Unlike other methods depending on the depth range and differentiable homograph warping to build the 3D cost volume, we use a network to match pixels along the epipolar line and triangulate the depth. Therefore, we first discuss the relationship between the flow, the depth, and the E-flow. Then, we extend the E-flow to multi-view and explain the details of DispMVS and the loss function.

Flow and Depth

Given a reference view \( r \) and a source view \( s \) with their interior matrix \( K_r, K_s \) and their relative exterior matrix \( [R_s, T_s] \), we define \( d_r, d_s \) as the depth, and \( f_{r \rightarrow s}, f_{s \rightarrow r} \) as the flow of each view. Assuming that the scene is static, we can convert depth to flow and vice versa according to the multi-view geometry (Hartley and Zisserman 2003).

Depth The depth describes the 3D shape of an image and can re-project a pixel on the image plane to 3D space. Eq. 1 re-projects a pixel \( p_r \) in \( r \) to \( P_{pr} \) in 3D by its depth \( d_r(p_r) \), in which \( p_r \) is the homogeneous representation of \( p_r \) for computation efficiency. \( P_{pr} \) can also be projected to \( s \) by Eq. 2.

\[
P_{pr} = d_r(p_r)K_r^{-1} \tilde{p}_r \tag{1}
\]

\[
p_s \simeq K_s[R_sP_{pr} + T_s] \tag{2}
\]

Flow The flow describes the movement of pixels on the image plane between two images. For a matched pixel pair \( p_r \) in \( r \) and \( p_s \) in \( s \), we calculate the flow \( \tilde{f}_{pr \rightarrow s} \) by Eq. 3. Generally, the flow does not need to follow geometry constraints and has two degrees of freedom.

\[
\tilde{f}_{r \rightarrow s}(p_r) = p_s - p_r \tag{3}
\]
**Depth to Flow** Eq. 4 shows how to convert the the \( d_r(p_r) \) to the \( f_r(p_r) \), where \( \Rightarrow \) denotes the conversion. We first re-project \( p_r \) to \( P_{ps} \) by \( d_r(p_r) \) as Eq. 1 shows and then project \( P_{ps} \) to \( s \) by Eq. 2 to get the matched pixel \( p_s \). Finally, we can calculate \( \hat{f}_{r \rightarrow s}(p_r) \) by Eq. 3.

\[
d_r(p_r) \Rightarrow P_{ps} \Rightarrow p_s \Rightarrow \hat{f}_{r \rightarrow s}(p_r) \tag{4}
\]

**Flow to Depth** Although triangulation is straightforward method to convert \( f_r \) to \( d_r \) (Hartley and Zisserman 2003), it has to solve a not differentiable homogeneous linear function. Considering this, we use a differentiable closed-form solution to calculate the depth, even though it is not optimal. Given \( p_r \) and \( f_{r \rightarrow s}(p_r) \), we can determine \( p_s \) by Eq. 3. Based on multi-view geometric consistency, we have the constrain in Eq. 5:

\[
d_r(p_r)K_r^{-1} \hat{p}_r = R_sp_sK_s^{-1}p_s + T_s \tag{5}
\]

Let \( T_s = (t_{sx}, t_{sy}, t_{sz})^T \), \( K_r^{-1} \hat{p}_r = (p_{rx}, p_{ry}, p_{rz})^T \) and \( R_sp_sK_s^{-1}p_s = (p_{sx}, p_{sy}, p_{sz})^T \). We can calculate \( d_r(p_r) \) by Eq. 6:

\[
\begin{align*}
    d_{xr}(p_r) &= (t_{sx}p_{sz} - t_{sz}p_{sx})/(p_{sx}p_{sz} - p_{sz}p_{sx}) \\
    d_{yr}(p_r) &= (t_{sy}p_{sz} - t_{sz}p_{sy})/(p_{sy}p_{sz} - p_{sz}p_{sy})
\end{align*} \tag{6}
\]

Eq. 6 shows that there are two ways to compute the depth, namely \( d_{xr}(p_r) \) and \( d_{yr}(p_r) \), since \( \hat{f}_{r \rightarrow s}(p_r) \) is a 2D vector that provides flow in \( x \) dimension \( f_{r \rightarrow x}(p_r) \) and \( y \) dimension \( f_{r \rightarrow y}(p_r) \). Theoretically, \( d_{xr}(p_r) \) equals \( d_{yr}(p_r) \). However, a smaller flow is not numerically stable and will bring noise into the triangulation. Therefore we select the depth triangulated by the larger flow by Eq. 7:

\[
d_r(p_r) = \begin{cases} 
    d_{xr}(p_r) & \text{if } |\hat{f}_{r \rightarrow x}(p_r)| \geq |\hat{f}_{r \rightarrow y}(p_r)| \\
    d_{yr}(p_r) & \text{if } |\hat{f}_{r \rightarrow x}(p_r)| < |\hat{f}_{r \rightarrow y}(p_r)| 
\end{cases} \tag{7}
\]

**E-flow: The Epipolar Disparity Flow**

The flow describes the movement of a pixel on the image plane but does not obey the epipolar geometry, which introduces ambiguity when triangulating the depth. Therefore, we use the epipolar geometry to restrain the flow and define the epipolar disparity flow (E-flow) by Eq. 8, where \( \hat{e}_{dir} \) is the normalized direction vector of the epipolar line, and \( \cdot \) is the dot product of vectors.

\[
ed_{r \rightarrow s}(p_r) = \hat{e}_{dir}(p_r) \cdot (p_s - p_r) \tag{8}
\]

**E-flow and Flow** Compared with the flow, the E-flow is a scalar and only moves on the epipolar line. In the static scene, the flow and the E-flow are two different ways to describe pixel movement, and their relationship is shown in Eq. 9. Figure 3 visualizes the flow \( f_{r \rightarrow s}(p_r) \) and the E-flow \( e_{r \rightarrow s}(p_r) \) of pixel \( p_r \).

\[
f_{r \rightarrow s}(p_r) = \hat{e}_{dir}(p_r)e_{r \rightarrow s}(p_r) \tag{9}
\]

**E-flow and Depth** Considering the relationship between the E-flow and the flow, and the relationship between the flow and the depth, we can convert E-flow to depth, and vice versa by Eq. 10, in which \( \Leftrightarrow \) denotes the interconversion.

\[
e_{r \rightarrow s} \Leftrightarrow f_{r \rightarrow s} \Leftrightarrow d_r \tag{10}
\]

**DispMVS**

Given a reference image \( r \) and \( N \) source images \( s_i (1 \leq i \leq N) \), DispMVS firstly extracts features from all input images and then iteratively updates the depth from random initialization. DispMVS separately estimates the E-flow of each pair by a 2D cost volume and converts the E-flow to the depth for later multi-view depth fusion by a weighted sum. Figure 2 shows the pipeline of DispMVS with the first iteration at the coarse stage.

**Feature Extraction** Following RAFT (Teed and Deng 2020), we use two identical encoders to extract features from input images. One encoder simultaneously extracts matching features from input images to calculate similarities. Meanwhile, another encoder extracts context features from the reference image to iteratively update the E-flow. As we apply a coarse-to-fine strategy to speed up efficiency and accuracy, these encoders use a UNet structure (Ronneberger, Fischer, and Brox 2015) to extract coarse feature maps \( c_r, c_{si} \) with resolution 1/16 for the coarse stage and \( f_r, f_{si} \) with resolution 1/4 for the fine stage. Besides, we also use the deformable convolutional network (Dai et al. 2017) at the decoder part to capture valuable information.

**Initialization of E-flow** DispMVS relies on the E-flow to estimate the depth, so it needs an initial depth as the starting point. We adopt the initialization strategy from PatchMatch.
tend the E-flow to the multi-view situation. DispMVS con-
plies to two views, DispMVS utilizes a weighted sum to ex-
the coarse stage and weight to the GRU module to estimate a new e-
flow, thus further improving multi-view consistency.

There are several conversion between the depth and the E-
similarity. Figure 4 compares two different sampling spaces.

DispMVS reconstructs a coarse depth from a random depth.

With \( d_{r,0} \), DispMVS can initialize the E-flow for pair by Eq.
10, which is zero for the first iteration. Figure 2 shows how
DispMVS initializes the E-flow for pair.

---

Coarse-to-Fine Strategy Following CasMVSNet (Gu et al. 2020), DispMVS uses a coarse-to-fine strategy. Disp-
MVS uses \( c_r, c_s, f_r \) with \( t_c \) iterations at the coarse stage and
\( f_r, f_s \) with \( t_f \) iterations at the fine stage. DispMVS starts
from a random depth at the coarse stage and upsamples (Teed and Deng 2020) the coarse depth to the fine stage for
later refinement. Generally, DispMVS needs more \( t_c \) and
fewer \( t_f \) to improve efficiency and accuracy.

Loss Function As DispMVS outputs a depth in each iter-
ation, we calculate the L1-loss for all depth maps to speed up
convergence. To improve the stability of the training proce-
sure, we use the inverse depth range to normalize the ground
truth (gt) and the depth \( d_{r,i} \). Eq. 14 shows our loss function,
in which \( \gamma = 0.9 \):

\[
loss = \sum_{j=t_c, t_f} \sum_{0 < i < j} \gamma^i \| \text{norm}(gt_i) - \text{norm}(d_{r,i}) \| \tag{14}
\]

---

Experiments

In this section, we benchmark our DispMVS on two public
datasets and compare it with a set of existing methods. We
also conduct ablation experiments to explore the effects of
different settings of DispMVS.

Datasets We use three different datasets throughout our
experiments. The DTUMVS (Aanæs et al. 2016) is an indoor
data set in a controlled environment containing 79 scenes for
training, 22 for testing, and 18 for validation. The Blended-
MVS (Yao et al. 2020) is a large dataset captured from various
outdoor scenes, with 106 scenes for training and the rest
7 scenes for testing. The Tanks&Temple (Knapitsch et al.
2017) is an outdoor multi-view stereo benchmark that contains
14 real-world scenes under complex conditions.

Implementation We implement DispMVS by PyTorch
(Paszke et al. 2019) and train two models on the DTUMVS
and the BlendedMVS separately. On the DTUMVS, we set
the image resolution to \( 640 \times 512 \) and \( N = 5 \). On the
BlendedMVS, we set the image resolution to \( 768 \times 576 \)
and \( N = 5 \). For all models, we apply the training strategy in
PatchmatchNet (Wang et al. 2021) for better learning.
epochs. The training procedure is finished on two V100 with
an initial learning rate of 0.0002 that halves every four epochs for 16
epochs. The training procedure is finished on two V100 with
an initial learning rate of 0.0002 that halves every four epochs for 16
epochs. The training procedure is finished on two V100 with
an initial learning rate of 0.0002 that halves every four epochs for 16
each inference is smaller than $1e^{-6}$, and evaluate point clouds. Table 4 shows that the variance
more GPU memory than DispMVS. Overall, the results on
Tanks&Temple demonstrate that DispMVS has robust
generation of a depth range as it only constructs a 2D cost vol-
sum on the image plane along the epipolar line. We select
differentiable homography. However, DispMVS is insensi-
we visualize some point clouds generated by DispMVS on DTUMVS (Aanæs et al. 2016) and Tanks & Temple (Knapitsch et al. 2017).

**Ablation Studies**
In this subsection, we discuss the core parts of our method. Considering that the RAFT structure has been thoroughly
studied in (Teed and Deng 2020; Lipson, Teed, and Deng 2021; Wang et al. 2022), we conduct ablation experiments
on the coarse-to-fine strategy, the random initialization, and the changes in depth range. Throughout all ablation experi-
ments, we use the DTUMVS as a baseline dataset.

The Coarse-to-Fine Strategy DispMVS-M firstly estimates the depth with the feature from 1/16 and refines the depth with the feature from 1/4. DispMVS-S only extrac-
t features from 1/8 to recover the depth map to make the comparison fairer. Table 3 shows that the coarse-to-fine strat-
egy dramatically improves the overall score from 0.455 for DispMVS-M to 0.339 for DispMVS-S. Therefore, we
choose DispMVS-M as the method used in this paper.

Random Initialization Unlike existing methods that build the 3D cost volume from a known depth range, Disp-
MVS starts from a random initial depth, which means that the input of DispMVS could be different every time. To
measure the effects of the random initialization, we conduct three times of initialization without fixing the random seed
and evaluate point clouds. Table 4 shows that the variance of metrics between different inferences is smaller than $1e^{-6}$, which proves that DispMVS are robust to the random initial depth and can always generate a high-quality depth map.

Depth Range The existing MVS methods split a given depth range into several bins and build a 3D cost volume by
differentiable homography. However, DispMVS is insensitive to the depth range as it only constructs a 2D cost vol-
ume on the image plane along the epipolar line. We select

![Figure 5: Point clouds. We visualize some point clouds generated by DispMVS on DTUMVS (Aanæs et al. 2016) and Tanks & Temple (Knapitsch et al. 2017).](image)

**Evaluation on DTUMVS**
We evaluate the DTUMVS on the test part and resize all images to $1600 \times 1152$ with $N = 5$. Table 1 compares Disp-
MVS with other state-of-the-art methods. We split existing
methods into traditional methods, 3D convolution-based methods, and RNN-based methods. DispMVS has the best
overall score among RNN-based methods and is 0.024 lower than IterMVS (Wang et al. 2022) and 0.018 lower than AA-
RMVSNet (Wei et al. 2021). DispMVS ranks the 3rd among all methods. GBiNet (Mi, Di, and Xu 2022) and UniMVS-
Net (Peng et al. 2022) are the top two methods, but they incur much higher GPU memory. We visualize some point
clouds of DTUMVS generated by DispMVS in the first row
of Figure 5. These qualitative and quantitative experimental results demonstrate the effectiveness of DispMVS in obtain-
ing depth by triangulation, even though DispMVS does not construct any 3D cost volume.

**Evaluation on Tanks&Temple**
As Tanks&Temple does not provide training samples, we apply the model pretrained on the BlendedMVS to it. We

| Method          | T | Acc↓ | Comp↓ | Overall↓ | Mem↓ |
|-----------------|---|------|-------|----------|------|
| Campbell et al. (2008) | 0.835 | **0.554** | 0.695 | — | — |
| Furuikawa et al. (2009) | 0.613 | 0.941 | 0.777 | — | — |
| Tola et al. (2012) | 0.342 | 1.190 | 0.766 | — | — |
| Galliani et al. (2015) | **0.283** | 0.873 | **0.578** | — | — |
| Yao et al. (2018) | 0.396 | 0.527 | 0.462 | 9384M | — |
| Chen et al. (2019) | 0.342 | 0.411 | 0.376 | — | — |
| Luo et al. (2019) | 0.406 | 0.434 | 0.420 | — | — |
| Yu et al. (2020) | 0.336 | 0.403 | 0.370 | — | — |
| Yang et al. (2020) | **0.296** | 0.406 | 0.351 | — | — |
| Zhang et al. (2020) | 0.369 | 0.361 | 0.365 | 4775M | — |
| Xu et al. (2020a) | 0.417 | 0.437 | 0.427 | — | — |
| Gu et al. (2020) | 0.325 | 0.385 | 0.355 | 4591M | — |
| Cheng et al. (2020) | 0.338 | 0.349 | 0.344 | 4057M | — |
| Ma et al. (2021) | 0.414 | 0.297 | 0.355 | — | — |
| Peng et al. (2022) | 0.352 | 0.278 | 0.315 | 3216M | — |
| Mi et al. (2022) | 0.315 | **0.262** | **0.289** | **2018M** | — |

| Method          | R | Acc↓ | Comp↓ | Overall↓ | Mem↓ |
|-----------------|---|------|-------|----------|------|
| Yao et al. (2019) | 0.383 | 0.452 | 0.417 | — | — |
| Yan et al. (2020) | 0.395 | 0.378 | 0.386 | — | — |
| Wei et al. (2021) | 0.376 | 0.339 | 0.357 | 11973M | — |
| Wang et al. (2021) | **0.427** | **0.277** | 0.352 | 1629M | — |
| Wang et al. (2022) | 0.373 | 0.354 | 0.363 | **842M** | — |
| ours             | **0.354** | 0.324 | **0.339** | 1368M | — |

Table 1: The evaluation results on DTUMVS (Aanæs et al. 2016). The lower the Accuracy (Acc), Completeness (Comp), Overall and Mem (GPU Memory), the better. We split methods into three categories and highlight the best in bold for each, where T means traditional methods, C means cost-volume methods, and R means RNN-based methods.
two state-of-the-art methods (GBiNet (Mi, Di, and Xu 2022) and IterMVS (Wang et al. 2022)) and manually change the depth range by Eq. 15. All methods are trained by the same dataset with the same depth range. Table 5 shows that performance of GBiNet and IterMVS decreases dramatically, but DispMVS can be robust to these changes. Figure 1 visualizes point clouds generated by different methods with different depth ranges, where GBiNet and IterMVS cannot converge when the depth range is too large.

\[
\text{Comp}_x = \begin{cases} 
  d_{\text{min}} = d_{\text{min}} / x \\
  d_{\text{max}} = d_{\text{max}} \times x 
\end{cases}
\]  

(15)

Limitations

As DispMVS needs to keep building the 2D cost volume during the iteration, its computational efficiency is relatively low. In our experiment, DispMVS needs around 0.7 seconds to process a view on the DTUMVS. Compared with IterMVS (Wang et al. 2022) which only needs around 0.3 seconds per view, DispMVS needs a more efficient epipolar matching module. In addition, DispMVS needs around 48G GPU memory during training because DispMVS needs several iterations to update the depth by the GRU module, which needs to save all gradients and intermediate results.

Table 2: The evaluation results on Tanks & Temple (Knapitsch et al. 2017). Higher scores indicate higher quality of the point cloud. We split methods into two categories and highlight the best in bold for each.

| Method | Mean | Aud. | Bal. | Cou. | Mus. | Pal. | Tem. |
|--------|------|------|------|------|------|------|------|
| ours   | 34.90| 26.09| 33.19| 44.90| 28.49| 38.75|
| ours   | 34.90| 26.09| 33.19| 44.90| 28.49| 38.75|

Table 3: Comparison between single stage and multi stage. “-S” indicates reconstruction with only one resolution at 1/8, while “-M” indicates reconstruction with multiple resolutions (the method in this paper).

| Method | Acc↓ | Comp↓ | Overall↓ |
|--------|------|-------|----------|
| DispMVS-M | 0.354 | 0.324 | 0.339 |
| DispMVS-S | 0.500 | 0.410 | 0.455 |

Table 4: Evaluate the random initialization. A lower variance means the difference between multi results is smaller.

| Method | Acc↓ | Comp↓ | Overall↓ |
|--------|------|-------|----------|
| rand-1 | 0.353829 | 0.324110 | 0.338970 |
| rand-2 | 0.354811 | 0.324946 | 0.339878 |
| rand-3 | 0.354272 | 0.324324 | 0.339298 |
| variance | 2.418e-7 | 1.890e-7 | 2.110e-7 |

Table 5: Influences of changing the depth range. The lower, the better for all metrics under different depth ranges.

| Range | Method | Acc↓ | Comp↓ | Overall↓ |
|-------|--------|------|-------|----------|
| r_{ng1} | Mi et al. (2022) | 0.315 | 0.262 | 0.289 |
|        | Wang et al. (2022) | 0.373 | 0.354 | 0.363 |
|        | ours | 0.354 | 0.324 | 0.339 |
| r_{ng2} | Mi et al. (2022) | 0.480 | 0.556 | 0.518 |
|        | Wang et al. (2022) | 0.532 | 1.471 | 1.002 |
|        | ours | 0.348 | 0.404 | 0.376 |
| r_{ng3} | Mi et al. (2022) | 0.618 | 1.303 | 0.960 |
|        | Wang et al. (2022) | 0.935 | 6.985 | 3.960 |
|        | ours | 0.314 | 0.671 | 0.493 |

Conclusion

This paper introduces a new pipeline of MVS, called DispMVS, which does not need to build any 3D cost volumes but triangulates the depth map by multi-view geometry. DispMVS is a depth range invariant method and can be generalized to the dataset with different ranges with the training set, which proves that 3D cost volume is unnecessary for MVS. Compared with existing learning-based methods, DispMVS lets the network focus on matching pixels that the CNN network is good at and uses the multi-view geometry to deal with the geometry information. Experiments on datasets show that DispMVS achieves comparable results with other 3D convolution methods and outperforms RNN-based methods with a lower GPU memory requirement.

Acknowledgments

This work was supported in part by grant RMGS2021-8-10 from Hong Kong Research Matching Grant Scheme and the NVIDIA Academic Hardware Grant.
References

Aanæs, H.; Jensen, R. R.; Vogiatzis, G.; Tola, E.; and Dahl, A. B. 2016. Large-scale data for multiple-view stereopsis. *International Journal of Computer Vision*, 120(2): 153–168.

Bleyer, M.; Riehmann, C.; and Rother, C. 2011. Patchmatch stereo-stereo matching with slanted support windows. In *Bmvc*, volume 11, 1–11.

Campbell, N. D.; Vogiatzis, G.; Hernández, C.; and Cipolla, R. 2008. Using multiple hypotheses to improve depth-maps for multi-view stereo. In *European Conference on Computer Vision*, 766–779. Springer.

Chang, D.; Božič, A.; Zhang, T.; Yan, Q.; Chen, Y.; Süsstrunk, S.; and Nießner, M. 2022. RC-MVSNet: Unsupervised Multi-View Stereo with Neural Rendering. In *European conference on computer vision*.

Chen, R.; Han, S.; Xu, J.; and Su, H. 2019. Point-based multi-view stereo network. In *Proceedings of the IEEE/CVF international conference on computer vision*, 1538–1547.

Cheng, S.; Xu, Z.; Zhu, S.; Li, Z.; Li, L. E.; Ramamoorthi, R.; and Su, H. 2020. Deep stereo using adaptive thin volume representation with uncertainty awareness. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2524–2534.

Dai, J.; Qi, H.; Xiong, Y.; Li, Y.; Zhang, G.; Hu, H.; and Wei, Y. 2017. Deformable convolutional networks. In *Proceedings of the IEEE international conference on computer vision*, 764–773.

Dai, Y.; Zhu, Z.; Rao, Z.; and Li, B. 2019. Mvs2: Deep unsupervised multi-view stereo with multi-view symmetry. In *2019 International Conference on 3D Vision (3DV)*, 1–8. Ieee.

Dosovitskiy, A.; Beyer, L.; Kolesnikov, A.; Weissenborn, D.; Zhai, X.; Unterthiner, T.; Dehghani, M.; Minderer, M.; Heigold, G.; Gelly, S.; et al. 2021. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*.

Furukawa, Y.; and Ponce, J. 2009. Accurate, dense, and robust multiview stereopsis. *IEEE transactions on pattern analysis and machine intelligence*, 32(8): 1362–1376.

Galliani, S.; et al. 2015. Massively parallel multiview stereopsis by surface normal diffusion. In *Proceedings of the IEEE International Conference on Computer Vision*, 873–881.

Gu, X.; Fan, Z.; Zhu, S.; Dai, Z.; Tan, F.; and Tan, P. 2020. 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*, 2495–2504.

Hartley, R.; and Zisserman, A. 2003. *Multiple view geometry in computer vision*. Cambridge university press.

Huang, B.; Yi, H.; Huang, C.; He, Y.; Liu, J.; and Liu, X. 2021. M3VSNNet: Unsupervised multi-metric multi-view stereo network. In *2021 IEEE International Conference on Image Processing (ICIP)*, 3163–3167. Ieee.

Khot, T.; Agrawal, S.; Tulsiani, S.; Mertz, C.; Lucey, S.; and Hebert, M. 2019. Learning unsupervised multi-view stereopsis via robust photometric consistency. *arXiv preprint arXiv:1905.02706*.

Kingma, D. P.; and Ba, J. 2015. Adam: A Method for Stochastic Optimization. In *International Conference on Learning Representations*.

Knapitsch, A.; Park, J.; Zhou, Q.-Y.; and Koltun, V. 2017. Tanks and temples: Benchmarking large-scale scene reconstruction. *ACM Transactions on Graphics (ToG)*, 36(4): 1–13.

Kostrikov, I.; Horbert, E.; and Leibe, B. 2014. Probabilistic labeling cost for high-accuracy multi-view reconstruction. In *Proceedings of the ieee conference on computer vision and pattern recognition*, 1534–1541.

Lhuillier, M.; and Quan, L. 2005. A quasi-dense approach to surface reconstruction from uncalibrated images. *IEEE transactions on pattern analysis and machine intelligence*, 27: 418–433.

Lipson, L.; Teed, Z.; and Deng, J. 2021. Raft-stereo: Multilevel recurrent field transforms for stereo matching. In *2021 International Conference on 3D Vision (3DV)*, 218–227. Ieee.

Liu, J.; and Ji, S. 2020. A novel recurrent encoder-decoder structure for large-scale multi-view stereo reconstruction from an open aerial dataset. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 6050–6059.

Luo, K.; Guan, T.; Ju, L.; Huang, H.; and Luo, Y. 2019. P-mvsnets: Learning patch-wise matching confidence aggregation for multi-view stereo. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 10452–10461.

Luo, K.; Guan, T.; Ju, L.; Wang, Y.; Chen, Z.; and Luo, Y. 2020. Attention-aware multi-view stereo. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 1590–1599.

Ma, X.; Gong, Y.; Wang, Q.; Huang, J.; Chen, L.; and Yu, F. 2021. EPP-MVSNet: Epipolar-assembling based Depth Prediction for Multi-view Stereo. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 5732–5740.

Mi, Z.; Di, C.; and Xu, D. 2022. Generalized Binary Search Network for Highly-Efficient Multi-View Stereo. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 12991–13000.

Mildenhall, B.; Srinivasan, P. P.; Tancik, M.; Barron, J. T.; Ramamoorthi, R.; and Ng, R. 2020. Nerf: Representing scenes as neural radiance fields for view synthesis. In *European conference on computer vision*, 405–421. Springer.

Paszke, A.; Gross, S.; Massa, F.; Lerer, A.; Bradbury, J.; Chanan, G.; Killeen, T.; Lin, Z.; Gimelshein, N.; Antiga, L.; et al. 2019. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32.
