Adaptive depth estimation for pyramid multi-view stereo

Jie Liao, Yanping Fu, Qingan Yan, Fei Luo, Chunxia Xiao

School of Computer Science, Wuhan University, Wuhan, China
School of Computer Science and Technology, Anhui University, Hefei, China
Silicon Valley Research Center for Multimedia Software, JD.com, Beijing, China

ABSTRACT

In this paper, we propose a Multi-View Stereo (MVS) network which can perform efficient high-resolution depth estimation with low memory consumption. Classical learning-based MVS approaches typically construct 3D cost volumes to regress depth information, making the output resolution rather limited as the memory consumption grows cubically with the input resolution. Although recent approaches have made significant progress in scalability by introducing the coarse-to-fine fashion or sequential cost map regularization, the memory consumption still grows quadratically with input resolution and is not friendly for commodity GPU. Observing that the surfaces of most objects in real world are locally smooth, we assume that most of the depth hypotheses upsampled from a well-estimated depth map are accurate. Based on the assumption, we propose a pyramid MVS network based on the adaptive depth estimation, which gradually refines and upsamples the depth map to the desired resolution. Instead of estimating depth hypotheses for all pixels in the depth map, our method only performs prediction at adaptively selected locations, alleviating excessive computation on well-estimated positions. To estimate depth hypotheses for sparse selected locations, we propose the lightweight pixelwise depth estimation network, which can estimate depth value for each selected location independently. Experiments demonstrate that our method can generate results comparable with the state-of-the-art learning-based methods while reconstructing more geometric details and consuming less GPU memory.

1. Introduction

Given images calibrated manually or through Structure-From-Motion (SFM) algorithms [2–6,34], Multi-View Stereo reconstructs a dense representation of the target scene. The reconstruction results can be applied in automatic geometry, scene classification, image-based rendering, and robot navigation. The publishing of the MVS training dataset [7,8] facilitates the design of learning-based MVS algorithms and boosts the progress in this field.

Currently, learning-based MVS methods have achieved excellent performance on the MVS benchmarks [8–10]. Instead of utilizing handcrafted similarity measuring metrics and engineered cost volume regularization, learning-based MVS methods learn to extract features, measure the similarity across images, and regularize the cost volumes. The learned ones have larger perception domains and can utilize high-level semantic information. Therefore, learning-based depth estimation is relatively immune to specular, low-textured, and reflective regions that are intractable for traditional methods. Classical learning-based MVS algorithms [11,12] build 3D cost volumes and regularize them with 3D CNNs. Although the 3D cost volume and 3DCNNs are inherently suitable for handling 3D information, the corresponding GPU memory consumption is large, leading to the methods based on them limited to low-resolution input images and output results. To improve the scalability of the network for MVS, R-MVSNet [1] sequentially regularizes the 2D cost maps along the depth direction via the gated recurrent unit (GRU), leading to only quadratic GPU memory growth with volume resolution. Point-MVSNet [13] introduces a point-based deep iterative framework and reduces the computational cost by concentrating only on depth range near the output point cloud from the last iteration. Although both [1] and [13] make a great contribution to the scalability improvement of learning-based MVS methods, their methods still only generate down-sampled depth maps due to limited GPU memory. Yang et al. [14] and Gu et al. [15] propose to estimate depth maps in the coarse-to-fine manner. Instead of constructing a cost volume at a fixed resolution, their method...
ods build cost volume pyramids. The coarsest cost volume in the pyramid architecture is built upon sparse sampled front-parallel planes across the whole depth range. The subsequential cost volumes are adaptively constructed and narrowed in depth direction based on the depth map estimated from the previous level. By avoiding unnecessary estimation on insignificant depth values and concentrating on more meaningful regions, their method significantly reduces memory and time consumption while generating better depth maps with the same resolution as the input image. However, while addressing 2-megapixel images, both methods require around 10GB GPU RAM and their memory consumption grows quadratically with increasing output resolution. Considering that present mainstream cameras and smartphones easily output images with over 2 megapixels and the memories of most GPUs are less than 8GB, higher memory efficiency is required for the learning-based MVS method to be applicable in our daily lives.

In this paper, we also adopt the pyramid architecture to perform the coarse-to-fine depth estimation. However, we conduct the estimation from a completely different perspective for avoiding excessive computation on unnecessary regions. Instead of estimating depth values for all pixels at each level of the pyramid, we only refine depth values at a small set of adaptively selected locations. The key observation is that as the surfaces of target objects in the real-world are often continuous, depth hypotheses belong to the same object, especially artificial objects, are usually locally smooth. As a result, most depth hypotheses upsampled from a well-estimated depth map are relatively convincing. Erroneous interpolated depth values are mainly distributed around object boundaries or areas with high-frequency geometry, which only makes up a small proportion of the input images. Based on this observation, we introduce the adaptive depth prediction network for MVS, which can perform efficient high-resolution depth map estimation by iteratively upsampling the depth map and refine the depth values only at adaptively selected locations (see Fig. 1).

We first preprocess each input image with R-MVSNet to generate an initial depth map. In our approach, we build an image pyramid for each input image. At each level of the pyramid architecture, we adaptively select locations where depth hypotheses are most likely to be inaccurate for further refinement. For each selected location, we construct the depth candidates and cost vectors according to geometry estimated from related views in the previous level and perform pixelwise depth estimation using sequential regularizaton. As we perform depth estimation for only a small number of locations with a short search range at each level, our method can effectively alleviate allocating unnecessary computational resources on well-estimated regions. The result is that our method can generate competitive results compared to state-of-the-art while utilizing much less GPU memory and reconstructing more geometric details.

While it is noteworthy that we share the similar insight with [16] as introducing the pyramid architecture to gradually refine the depth map at locations where depth hypotheses are inconsistent with related views, our work differs from theirs in the following three main aspects. First, PVA-MVSNet proposed in [16] performs depth estimation throughout the depth range for the whole images at multiple scales. By contrast, our method only performs depth estimation at adaptively selected locations where depth hypotheses are most likely to be incorrect. Second, PVA-MVSNet only utilizes the estimated depth hypotheses from related views to check the geometric consistency of depth hypotheses estimated in the reference view. Our method further warps depth maps from related views to the reference view to provide additional geometric information for depth estimation. Finally, PVA-MVSNet learns weights for different pixels in the image or voxels in 3D space to construct the cost volumes, which can be illustrated as attention mechanism. Instead, our method learns weights for different views while constructing the cost vectors, which is the learned counterpart of view selection in traditional MVS. The main contributions of our paper are summarized as follows:

- We introduce the idea of iterative adaptive depth estimation for learning-based MVS to avoid excessive computation on well-estimated regions, which dramatically reduces the growth of computational cost and memory consumption with model resolution from quadratic to logarithm.
- We propose the cross pyramid architecture to aggregate multi-view depth information for prediction in the reference view. By calculating the average reprojection error map between the reference view and the related views, we select the locations that are likely wrongly estimated for further refinement. Warping the depth maps from associated views to the reference view provides additional indicative depth information for better geometry estimation.
- We propose the pixelwise depth estimation module which can estimate depth value for a single location independently. It can be utilized for sparse depth estimation and is more friendly to geometric detail reconstruction.

We name our method as ADEP-MVSNet (Adaptive Depth Estimation for Pyramid MVSNet) and validate the effectiveness of ADEP-MVSNet on the Tanks and Temples benchmark. The experiments demonstrate that ADEP-MVSNet can generate comparable

---

**Fig. 1.** Depth map estimation with ADEP-MVSNet. The initial depth map is estimated through R-MVSNet [1]. ADEP-MVSNet adaptively estimates the depth values at locations selected via our location selection strategy.

(a) Source image (b) Depth map by R-MVSNet (c) Location selection (d) Depth map by ADEP-MVSNet
results with the state-of-the-art learning-based MVS while reconstructing more geometric details with less GPU memory.

2. Related work

2.1. Traditional MVS

According to output representations, traditional MVS methods can be classified into three categories: 1) direct point cloud reconstructions [17,18], 2) volumetric reconstructions [19-21] and 3) depth map reconstructions [22-28].

Point-based methods directly operate on and output point clouds. To densify the point cloud, these methods usually adopt propagation strategies to propagate a well-optimized point to nearby 3D space. The drawback is that the propagation procedure is performed in a sequence, which is not friendly to parallel computation and takes a long time. Volumetric methods divide 3D space into regular grids and reconstruct the target scene by judging whether the voxels adhere to the target scene surface. The limitation of volumetric presentation is the inherent space discretization error and high memory consumption. In contrast, the depth map presentation is much more flexible. It breaks the complex MVS problem into relatively more tractable per-view depth estimation problems and can be easily fused into point cloud or volumetric presentation. According to MVS benchmarks [9,10], current state-of-the-art MVS methods [26-29] are all depth-map based. However, although the depth map estimation procedure can be parallelized, depth map based methods usually take a longer time. The reason is that the depth map estimation is performed for each view, which is analogous to estimating geometry as many times as the number of input images. Besides, most of the traditional MVS algorithms mainly estimate geometry by maximizing the Normalized Cross-Correlation (NCC) between the projections of the patch on the reference and related views. The photometric consistency based on NCC is fragile to textureless and specular surfaces, making the reconstruction of the corresponding regions hard work for traditional MVS algorithms.

2.2. Learning-based MVS

Recently, learning-based MVS approaches have demonstrated great potentials on various MVS benchmarks [8-10]. Instead of handcrafted feature extraction and engineered matching metrics, learning-based methods utilize learned ones that are more robust for illumination change and non-Lambertian surface. Similar to traditional MVS methods, learning-based methods can also be classified into three categories according to output representations: 1) volumetric reconstructions [11,30], 2) depth (disparity) map reconstructions [1,12,14,15,31] and 3) direct point cloud reconstruction [13].

Earlier learning-based MVS methods like SurfaceNet [30] and LSM [11] are volumetric based methods. SurfaceNet [30] learns the probability of voxels lying on the surface. LSM [11] presents a learnable system to back-project pixel features to the 3D volume and classify whether a voxel is occupied or not by the surface. Both SurfaceNet and LSM made a groundbreaking contribution to learning-based MVS. However, the volumetric representations are memory expensive, making the algorithms based on them limited to small-scale scenes.

[1,12,14-16,31] estimate the depth map for each view. DeepMVS [31] pre-warp the multi-view images to 3D space and uses deep networks for regularization and aggregation. MVSNet [12] proposes to learn the depth map for each view by constructing a cost volume followed by 3D CNN regularization. As the 3D cost volume requires GPU memory that is cubic to the input resolution, the scalability of MVSNet is rather limited. R-MVSNet [1] improves the scalability of MVSNet by sequentially regularizing the 2D cost maps, but the memory consumption still restricts the method to only output downsampled depth maps. PVA-MVSNet [16] constructs the pyramid architecture to aggregate the reliable depth estimation at the coarser scale to fill in the mismatched regions at the finer scale and utilizes the self-adaptive view aggregation to improve the reconstruction quality. However, the memory cost of PVA-MVSNet is even larger than the seminal MVSNet [12]. Instead of building 3D cost volumes at fixed resolution, CVP-MVSNet [14] and CasMVSNet [15] construct the cost volume pyramid. By gradually narrow the cost volumes according to depth estimation from previous levels, both two methods avoid unnecessary computation and memory consumption on insignificant regions. As a result, CVP-MVSNet and CasMVSNet are able to output the full-resolution depth map with high-quality. However, the memory requirement is quadratic to the input resolution.

Point-MVSNet [13] introduces a novel point-based deep framework that estimates the 3D flow for each point to refine the point cloud. The idea of concentrating only on depth range near the output point cloud comes from the last iteration inspired both [15] and [14]. Although Point-MVSNet is more scalable than MVSNet, it still can not estimate full-resolution depth maps for images provided by the Tanks and Temples benchmark.

3. Methodology

The surfaces of objects in the real world are generally continuous, and most of them are smooth. This phenomenon is particularly evident in artificial scenes. Taking the desk as example, the majority of the desk surface is made up of planes and high-frequency geometry basically occurs at edges or some decorative accessories. Based on this observation, we assume that most of the depth values upsampled from a well-estimated depth map are fundamentally accurate, except for those near high-frequency areas like object boundaries. We validate our assumption on the BlendedMVS [7] as shown in Section 4.1.

The overview of our method is depicted in Fig. 2. The input calibrated images are first processed with R-MVSNet to generate initial low-resolution depth maps. Then we iteratively upsample and refine the depth maps. The refinement mainly consists of three steps: 1) selecting the locations to be refined, 2) constructing depth candidates for selected locations, and 3) performing pixelwise depth estimation and updating the upsampled depth maps.

We first introduce the cross pyramid architecture, which bridges the depth estimation across the related views. Current learning-based MVS algorithms perform depth estimation mainly according to the feature (photometric) information across multiple views. However, the geometric consistency, which is equally important in MVS and widely considered in traditional MVS algorithms, is ignored. We construct the cross pyramid architecture to allow depth refinement at the current level to utilize depth hypotheses of related views at the previous level. The cross pyramid architecture is the foundation of our location selection and depth candidate construction procedure.

Then we illustrate our location selection strategy, which aims to find falsely estimated or upsampled depth values. One direct and straightforward way to select locations to be refined is to upsample the probability map and pick the locations with low probability values. However, as we only refine part of the locations in the current depth map and adopt different numbers of depth candidate at each level, the denominators for probability normalization of refined locations are different from that of other locations. On the other hand, depth maps estimated at the previous level enable us to check the geometric consistency. The geometric consistency directly relates to the depth map fusion procedure and the quality of the output point cloud. Therefore, we propose the location
The overview of our proposed method. The input images are first preprocessed with R-MVSNet to generate initial depth maps. At each level of the pyramid architecture, feature maps are extracted from the input images. Pixelwise depth estimation is performed at locations selected at the previous level to refine the depth maps. Subsequently, the refined depth maps are upsampled and the locations to be refined in the next stage are selected with our location selection strategy. The upsample and refinement is iterated until the final depth maps reach the desired resolution. The overview of pixelwise depth estimation can be referred in Fig. 4.

Quantitative validation for our assumption that the majority of the depth hypotheses upsampled from well-estimated depth maps are accurate. The left image demonstrates the percentage of accurate depth predictions in the depth map estimated by R-MVSNet (original) and that upsampled by scale 2 (resized). The right image shows the percentage of decreased accurate depth predictions due to depth map upsampling. The x-coordinate of the two figures means the image identity in the test set provided by BlendedMVS [7].

The overview of the pixelwise depth estimation procedure. The feature vectors for the selected location are sampled according to the projection of the patch model at depth $d$. The cost vector is constructed as the learned weighted variance of the feature vectors. The cost vectors at different depth candidates are sequentially regularized by the three-level stacked GRU to form the probability vector. The Pixelwise Depth Estimation module is trained as a classification problem with the cross-entropy loss.
selection strategy based on the geometric consistency between the reference image and related views.

Finally, we present our pixelwise depth estimation module, which subsequently regularizes the cost vectors and estimates the depth value for a single pixel.

3.1. Cross pyramid architecture

Fig. 5 demonstrates the comparison between the pyramid architecture utilized in recent learning-based MVS algorithms and our proposed cross pyramid architecture. Both architectures aim to gradually upsample and refine the depth map to the desired resolution. In the pyramid architecture, the depth map at the non-top level is refined mainly according to the feature (photometric) information across related views. The candidate depth values are distributed around depth values estimated at the previous level. If the depth value estimated at the previous level is far from the real value, re-estimating a depth value around the wrong one is a waste of computation resources. In contrast, our cross pyramid architecture bridges the depth estimation of multiple related views. At each level of the cross pyramid architecture, feature extraction and depth estimation are sequentially performed for each view. While performing depth estimation for the view at the non-top level, it utilize depth hypotheses of related views. For example, the geometry estimated in the current view may be erroneous but is correctly predicted in the associated views. In this case, warping the depth hypotheses from related views to the current view can provide correct depth candidates that must not be acquired near current erroneous depth hypotheses.

3.2. Location selection and depth candidate construction

We select the locations to be refined based on the reprojection error, which reflects the geometric consistency across multiple views and is closely related with the depth map fusion procedure. Take the depth map of the view \(i\) at the \(l\) level \(D^l_j\) as example, we first calculate its reprojection error maps with other related views, and then average the reprojection error for each pixel on \(D^l_j\). While calculating the reprojection error map \(E^l_j\) for current depth map \(D^l_j\) with a related depth map \(D^l_i\), we are faced with the occlusion problem. To be specific, due to view changes, the region captured in \(D^l_i\) may be occluded in \(D^l_j\), resulting in the corresponding reprojection error unreliable and meaningless as demonstrated in Fig. 6(c). Such occluded regions should be masked off before averaging the reprojection errors. We adopt a simple but effective method to judge the occluded regions. We warp the depth maps \(D^l_i\) and \(D^l_j\) onto each other to form the warped depth maps \(\hat{D}^l_i\) and \(\hat{D}^l_j\), and compare the depth map and warped depth map on each view to find regions where warped depth surpasses the original depth by a certain threshold \(\sigma\). The warp procedure can be formulated as

\[
\lambda_j x_j = K^l_j \left[ R, R^{-1} \left( (K^l_i)^{-1} x^T D^l_i(x) - t \right) \right].
\] (1)

\(R, K\) and \(t\) represent the rotation, calibration, and translation of the camera respectively. \(x\) is the coordinate of the pixel on the depth map \(D^l_i\) with its depth value as \(D^l_i(x)\). \(x_j\) represents the 2D coordinate of projection of \(x\) on view \(j\) and \(\lambda_j\) is the corresponding warped depth value. If the warped depth value \(\lambda_j\) surpassed the original depth value \(D^l_i(x_j)\) by the threshold \(\sigma\), \(x\) is deemed as occluded in view \(j\) and the corresponding reprojection error \(R^l_j(x)\) will be ignored during the averaging procedure. After acquiring the reprojection error map, we choose \(k\) locations with the largest reprojection error for further depth refinement. Fig. 6 demonstrates the reprojection error map and location selection map calculated with and without occlusion detection. It can be observed that the reprojection error map generated with occlusion detection can better reflect the locations where depth values are falsely estimated.

Given selected locations, we need to construct depth candidates for each location, from which the refined depth value will be chosen. In [13–15], depth candidates are distributed near the depth value estimated at the previous stage. There exists a possibility that the depth value estimated at the previous stage is erroneous and far from the true value, thus resulting in the true value still outside all depth candidates. On the other hand, the corresponding geometry may be correctly estimated in other views. Therefore, we consider the depth values warped from other views as the depth candidates. We first warp depth maps from \(N - 1\) related views to the current view to form the initial \(N\) depth candidates. Then we uniformly sample \(m\) depth values centering on each initial depth candidate to form \(M = m \times N\) depth candidates.

It is noteworthy that due to the discretization of the depth map, there will be cases that multiple warped depth values fall on the same location, and for some locations there are no warped depth values. The situations are shown in Fig. 7. For the first case, we select the minimal depth value as the final warped depth. For the
second case, we mask such locations for depth candidate construction. All selected locations and the corresponding feature vectors and depth candidates will be passed to the Pixel Depth Estimation module for further depth refinement.

3.3. Pixelwise depth estimation

Given the selected location and the corresponding depth candidates, the Pixelwise Depth Estimation (PDE) module aims to choose the optimal depth value from the depth candidates. Similar to R-MVSNet [1], the PDE is basically a recurrent neural network embedded with encoded camera parameters. The overview of the PDE is demonstrated as Fig. 4.

First, we need to build a series of cost vectors upon the depth candidates for each selected location. A straightforward way to build a cost vector on a depth candidate is first to sample feature vectors cross multiple feature maps via Eq. 1 and then compute the variance of extracted feature vectors as the cost vector. However, pixelwise extracted feature vectors lack enough contextual information for depth estimation, although the feature vectors are calculated through multiple layers of CNNs. Besides, the variance-based cost vector assumes that different views contribute equally to the depth estimation procedure, while this assumption fails in case of the occlusions and reflections.

To provide more contextual information, instead of extracting single feature vectors for the current location, we aggregate neighboring feature vectors. The neighbors are selected as the four corners of a \( \mu \times \mu \) patch centering the selected location. While extracting feature information from other views, the selected location and the neighbors are projected on other views to sample feature vectors. The feature vectors deriving from the same view are subsequently concatenated as one feature vector, representing the feature information on each view with the current location and depth candidate.

Different from previous methods [1,12,14–16] that consider multiple views contribute equally to the depth estimation for the reference image, we think the weights of different views are different for pixelwise depth estimation due to capture conditions like occlusions and illuminations. For each candidate view except the reference image, we calculate the element-wise absolute difference between reference feature vector \( f_0 \) and the feature vector sampled from current view \( f_i \) as \( f'_i = |f_0 - f_i| \). We sequentially input \( \{f'_i\}_{i=1}^{N-1} \) into a GRU layer to generate a series of values and then calculate the exponent of each output value as the view selection weight \( w_i \) of each view for depth estimation. Finally, we calculate the cost vector as the weighted variance of the aggregated feature vectors as

\[
C = \frac{\sum_{i=0}^{N-1} w_i (f'_i - \bar{f})}{\sum_{i=0}^{N-1} w_i}
\]

(2)

where we set \( w_0 \) for reference image as Euler number \( e \).

After constructing the cost vectors for each selected location, we adopt a 3-layer stacked GRU to regularize the cost vectors sequentially. The output of each GRU layer will be used as the input to the next GRU layer, and the output channel numbers of the three layers are set to 16, 4, 1 respectively. The regularized cost vectors will finally go through a softmax layer to generate the probability vector representing the probability for each depth candidate.

3.4. Loss

For each selected pixel, we calculate the cross-entropy between the newly estimated depth probabilistic vector and the one-hot coding for the depth candidate nearest to the ground truth depth value. Then the training loss is defined as the average cross-entropy for all selected pixels.

3.5. Postprocessing and Fusion

The same to R-MVSNet [1], we also utilize the variational depth map refinement proposed in [1] to refine the depth map in a small depth range by enforcing the multi-view photometric consistency. We use the fusion method provided by COLMAP [29] to fuse the output depth maps.

4. Experiments

In this section, we first validate our assumption that most of the depth hypotheses upsampled from a well-estimated depth map are accurate on the BlendedMVS dataset. Then we detail the implementation of our method. After that, we evaluate and compare our method with state-of-the-art learning-based MVS methods [1,12–16] on the Tanks and Temples benchmark. Subsequently, we analyze the scalability and efficiency of our method and perform the ablation study. Finally, we demonstrate the limitation of our method. All the experiments in this section are conducted on a single machine with an Intel Xeon(R) CPU E5-2630, 128G RAM, and a GeForce RTX 2080 Ti.

4.1. Assumption validation

In this section, we validate the assumption that the majority of the depth hypotheses upsampled from a well-estimated depth map
is accurate. The quantitative validation is conducted on the low-resolution set of the BlendedMVS dataset provided by [7]. BlendedMVS is a large-scale MVS dataset for generalized multi-view stereo networks. The dataset contains 17k MVS training samples covering a variety of 113 scenes, including architectures, sculptures, and small objects. The images and provided ground truth depth maps in the low-resolution set are at the resolution of 576 × 768. We first train the R-MVSNet on the training set of the low-resolution BlendedMVS dataset. Then we utilize the trained R-MVSNet to estimate depth maps for the validation set in the dataset with 192 front-parallel planes. The validation dataset contains 7 scenes with total 914 images. We define the accuracy value for the estimated depth map as the percentage of the depth hypotheses that are within one depth interval to the ground truth depth value. The depth interval is the distance between two neighboring front-parallel planes. For every 10 images in the validation set provided by the BlendedMVS dataset, we record the accuracy values of the depth map estimated by R-MVSNet (original) and the depth map upsampled by factor 2 (resized), and plot them as the left image in Fig. 3. Meanwhile, we also plot the decreased accuracy values caused by upscaling in the right image in Fig. 3. It can be observed that depth values upsampled from well-estimated depth maps are mostly accurate as the decreased accuracy values are no larger than 1.86%.

### 4.2. Implementation

We train the feature extraction module and PDE on the DTU dataset [32] and low-resolution BlendedMVS dataset [7]. The DTU dataset contains over 100 scenes under seven different lightning conditions captured with a fixed camera trajectory. Thanks to the open-source project released by [112], we can directly use the modified DTU dataset for training. The depth maps are generated from ground truth point clouds through the method utilized in [112]. All training images are resized to $H \times W = 512 \times 640$. The resolution of the images and ground-truth depth maps in the low-resolution BlendedMVS dataset is 576 × 768. We first run the R-MVSNet on both datasets with the input view number as 5 to generate the initial depth maps. For the DTU dataset, the depth hypotheses are sampled from 425mm to 905mm with the depth number $d = 192$. As the DTU dataset modified by [112] provides GT depth maps at a resolution of 128 × 160, we resize the input images to 256 × 320 to make initial depth maps output by R-MVSNet at a resolution of 64 × 80. For the BlendedMVS dataset, we adopt the default depth sampling number as $d = 128$ and utilize the default input images to generate initial depth maps at a resolution of 144 × 192.

We upsample the initial depth maps estimated and select 10000 locations for each depth map to train PDE. Among the 10000 locations, 75% of them are selected through our location selection strategy as illustrated in Section 3.2 and the remainders are uniformly sampled across the upsampled depth maps. We experimentally set the coefficient for expanding the initial depth candidates as $m = 7$. The depth interval between neighboring expanded depth candidates as 0.5 times the initial depth interval, and the threshold $\sigma$ for occlusion detection as 10 times the initial depth interval. We set the patch size as $7 \times 7$ for neighboring feature aggregation. During training, we use Adam optimizer with $\beta_1 = 0.9$ and $\beta_2 = 0.999$. The learning rate is set to 0.001 with exponential decay of 0.9 for every 10k iterations. The batch size is fixed to 5.

#### 4.3. Benchmark performance

We validate the effectiveness of our method on the Tanks and Temples Benchmark. The Tanks and Temples benchmark provides both training data and testing data. Ground-truth (GT) is only provided for the training dataset, which allows parameter tuning. Reconstruction results submitted to the Tanks and Temples benchmark will be evaluated in three aspects as recall, precision, and F-score. The recall is computed as the percentage of points from GT which are within a certain distance $\tau$ from the model. The precision is computed as the percentage of points from the model which are within a distance $\tau$ from the GT. The F-score is the harmonic average of recall and precision. For a detailed description of the Tanks and Temples benchmark, it is suggested to refer to [9]. We take the depth maps estimated by R-MVSNet with parameters described in [1] as input. The level of the cross pyramid architecture is set as 3, which ensures the final output resolution is close to the resolution of input images. During depth map upsampling and refinement, we set the input number of images as $N = 10$. For each iteration, we select $S = 30,000$ locations for depth refinement and set the expansion number $m = 5$. We set the depth interval for each none-top level as 0.5 and 0.25 times the initial depth interval, which is calculated as the depth range divided by the depth plane number provided by [1]. We set the patch size as $7 \times 7$ for neighboring feature aggregation. We use the fusion method provided by COLMAP [29]. We set the minimal consistent pixel number as 5 and the maximal reprojection error as 0.5. As the threshold $\tau$ (mm) varies in different datasets in the Tanks and Temples benchmark, we set the maximal depth error as 0.66 × $\tau$.

Table 1 demonstrates the F-scores of our proposed method against published state-of-the-art learning-based MVS algorithms on test datasets. It can be observed that ADEP-MVSNet and CVMVSNet [15] rank the first for two of the eight scenes. CVP-MVSNet [14] ranks the first for three of the eight scenes. PVA-MVSNet [16] ranks the first for one of the eight scenes. While only refining part of the depth maps at each level of the pyramid and consuming much less GPU memory, our method generates competitive results compared to CVMVSNet and CVP-MVSNet which refine the whole depth map at each iteration. Meanwhile, Our method slightly outperforms PVA-MVSNet which also refines the depth hypotheses at the locations adaptively selected according to geometric inconsistency. Compared to R-MVSNet, ADEP-MVSNet significantly improves the mean F-score over the eight scenes. The reason lies in the high-resolution output benefiting from less memory consumption and adaptive depth refinement. For the M60 and Horse dataset as $m = 7$, the depth interval between neighboring expanded depth candidates as 0.5 times the initial depth interval, and the threshold $\sigma$ for occlusion detection as 10 times the initial depth interval. We set the patch size as $7 \times 7$ for neighboring feature aggregation. During training, we use Adam optimizer with $\beta_1 = 0.9$ and $\beta_2 = 0.999$. The learning rate is set to 0.001 with exponential decay of 0.9 for every 10k iterations. The batch size is fixed to 5.

### 4.3. Benchmark performance

We validate the effectiveness of our method on the Tanks and Temples Benchmark. The Tanks and Temples benchmark provides both training data and testing data. Ground-truth (GT) is only provided for the training dataset, which allows parameter tuning. Reconstruction results submitted to the Tanks and Temples benchmark will be evaluated in three aspects as recall, precision, and F-score. The recall is computed as the percentage of points from GT which are within a certain distance $\tau$ from the model. The precision is computed as the percentage of points from the model which are within a distance $\tau$ from the GT. The F-score is the harmonic average of recall and precision. For a detailed description of the Tanks and Temples benchmark, it is suggested to refer to [9]. We take the depth maps estimated by R-MVSNet with parameters described in [1] as input. The level of the cross pyramid architecture is set as 3, which ensures the final output resolution is close to the resolution of input images. During depth map upsampling and refinement, we set the input number of images as $N = 10$. For each iteration, we select $S = 30,000$ locations for depth refinement and set the expansion number $m = 5$. We set the depth interval for each none-top level as 0.5 and 0.25 times the initial depth interval, which is calculated as the depth range divided by the depth plane number provided by [1]. We set the patch size as $7 \times 7$ for neighboring feature aggregation. We use the fusion method provided by COLMAP [29]. We set the minimal consistent pixel number as 5 and the maximal reprojection error as 0.5. As the threshold $\tau$ (mm) varies in different datasets in the Tanks and Temples benchmark, we set the maximal depth error as 0.66 × $\tau$.

Table 1 demonstrates the F-scores of our proposed method against published state-of-the-art learning-based MVS algorithms on test datasets. It can be observed that ADEP-MVSNet and CVMVSNet [15] rank the first for two of the eight scenes. CVP-MVSNet [14] ranks the first for three of the eight scenes. PVA-MVSNet [16] ranks the first for one of the eight scenes. While only refining part of the depth maps at each level of the pyramid and consuming much less GPU memory, our method generates competitive results compared to CVMVSNet and CVP-MVSNet which refine the whole depth map at each iteration. Meanwhile, Our method slightly outperforms PVA-MVSNet which also refines the depth hypotheses at the locations adaptively selected according to geometric inconsistency. Compared to R-MVSNet, ADEP-MVSNet significantly improves the mean F-score over the eight scenes. The reason lies in the high-resolution output benefiting from less memory consumption and adaptive depth refinement. For the M60 and Horse

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Mean</th>
<th>Family</th>
<th>Francis</th>
<th>Horse</th>
<th>LightHouse</th>
<th>M60</th>
<th>Panther</th>
<th>Playground</th>
<th>Train</th>
</tr>
</thead>
<tbody>
<tr>
<td>r (mm)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MVSNet</td>
<td>43.48</td>
<td>55.99</td>
<td>28.55</td>
<td>25.07</td>
<td>50.79</td>
<td>53.96</td>
<td>50.86</td>
<td>47.90</td>
<td>34.69</td>
</tr>
<tr>
<td>R-MVSNet</td>
<td>48.40</td>
<td>69.96</td>
<td>46.65</td>
<td>32.59</td>
<td>42.95</td>
<td>51.88</td>
<td>48.80</td>
<td>52.00</td>
<td>42.38</td>
</tr>
<tr>
<td>Point-MVSNet</td>
<td>50.55</td>
<td>61.79</td>
<td>41.15</td>
<td>34.20</td>
<td>50.79</td>
<td>51.97</td>
<td>50.85</td>
<td>52.38</td>
<td>43.06</td>
</tr>
<tr>
<td>CVP-MVSNet</td>
<td>54.03</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>
</tr>
<tr>
<td>CasMVSNet</td>
<td>56.84</td>
<td>76.50</td>
<td>47.74</td>
<td>36.34</td>
<td>55.02</td>
<td>57.28</td>
<td>54.28</td>
<td>57.43</td>
<td>47.54</td>
</tr>
<tr>
<td>PVA-MVSNet</td>
<td>54.46</td>
<td>69.36</td>
<td>46.80</td>
<td>46.01</td>
<td>55.74</td>
<td>57.23</td>
<td>54.70</td>
<td>56.70</td>
<td>49.06</td>
</tr>
<tr>
<td>ADEP-MVSNet</td>
<td>53.22</td>
<td>68.40</td>
<td>51.24</td>
<td>34.66</td>
<td>60.79</td>
<td>47.34</td>
<td>52.88</td>
<td>57.72</td>
<td>52.28</td>
</tr>
</tbody>
</table>
dataset, ADEP-MVSNet falls behind the state-of-the-art methods as the pixelwise depth estimation module can not utilize the global contextual information for depth inference, which may worsen the estimation in textureless regions like the base of the horse in Fig. 9.

Fig. 8 demonstrates precision map comparisons between ADEP-MVSNet and state-of-the-art learning-based MVS algorithms. The precision map draws the reconstructed points and assigns them the colors according to their distances to the ground truth. As demonstrated by the legend in Fig. 8, the points that are closer to the ground truth will be assigned with the lighter color. It can be observed that ADEP-MVSNet strikes a balance between detail reservation and reconstruction completeness. Compared to state-of-the-art learning-based MVS methods, ADEP-MVSNet generates results with more precious edges, which is attributed to our location selection strategy and adaptive depth estimation. The reconstructed point clouds of the Tanks and Temples intermediate dataset are demonstrated in Fig. 9.

4.4. Scalability and Efficiency

Table 2 demonstrates the GPU memory requirement runtime comparisons between ADEP-MVSNet and state-of-the-art learning-based MVS algorithms. The GPU memory consumption and runtime data of CVP-MVSNet, CasMVSNet, Point-MVSNet, and PVA-MVSNet is directly depicted from the corresponding paper [13–16]. As some of the data is recorded based on different input resolutions, we calculate the memory utility (Mem-Util), which denotes the GPU memory consumption per 10000 pixels in the output depth map, to measure the scalability of the methods. The smaller Mem-Util indicates the higher scalability. We also calculate the runtime utility (Time-Util), which denotes the time con-
Table 2
Scalability and efficiency comparisons between ADEP-MVSNet and state-of-the-art learning-based MVS algorithms. The memory utility (Mem-Util) is calculated as the GPU memory consumption per output 10000 pixels. The runtime utility (Time-Util) is calculated as the time consumption per 100000 pixels in the output depth map.

<table>
<thead>
<tr>
<th>Method</th>
<th>Input Resolution</th>
<th>Output Resolution</th>
<th>GPU Mem (MB)</th>
<th>Mem-Util (MB)</th>
<th>Runtime (s)</th>
<th>Time-Util (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>MVSNet</td>
<td>1920 × 1072</td>
<td>480 × 268</td>
<td>15308</td>
<td>1189</td>
<td>2.76</td>
<td>12.50</td>
</tr>
<tr>
<td>R-MVSNet</td>
<td>1920 × 1080</td>
<td>480 × 270</td>
<td>6714</td>
<td>518</td>
<td>5.09</td>
<td>39.27</td>
</tr>
<tr>
<td>Point-MVSNet</td>
<td>1600 × 1152</td>
<td>800 × 576</td>
<td>13081</td>
<td>283</td>
<td>3.04</td>
<td>6.60</td>
</tr>
<tr>
<td>CVP-MVSNet</td>
<td>1600 × 1152</td>
<td>800 × 576</td>
<td>8795</td>
<td>47</td>
<td>1.72</td>
<td>0.93</td>
</tr>
<tr>
<td>CasMVSNet</td>
<td>1152 × 864</td>
<td>1152 × 864</td>
<td>5345</td>
<td>53</td>
<td>0.49</td>
<td>0.49</td>
</tr>
<tr>
<td>PVA-MVSNet</td>
<td>1600 × 1184</td>
<td>1600 × 1184</td>
<td>24870</td>
<td>131</td>
<td>0.98</td>
<td>0.51</td>
</tr>
<tr>
<td>ADEP-MVSNet</td>
<td>1920 × 1080</td>
<td>1920 × 1080</td>
<td>5088</td>
<td>24</td>
<td>1.12</td>
<td>0.54</td>
</tr>
</tbody>
</table>

Table 3
Ablation study based on F-score, recall and precision for the Ignatius dataset: with the baseline settings, with uniformly random selected locations instead of adaptive selected locations (RS), with single pyramid architecture instead of cross pyramid architecture (PA), with the increasing number of selected locations (the depth map is only upsampled without adaptive depth estimation when the number of selected locations S = 0), without the occlusion detection (w/o OD), with increasing occlusion detection threshold \( \sigma \), with increasing levels of the cross pyramid architecture, and with the increasing patch size \( \mu \).

<table>
<thead>
<tr>
<th>Base</th>
<th>RS</th>
<th>PA</th>
<th>( S = 0 )</th>
<th>( S = 5000 )</th>
<th>( S = 10,000 )</th>
<th>( S = 15,000 )</th>
<th>( S = 30,000 )</th>
<th>( S = 40,000 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>F-score</td>
<td>78.2</td>
<td>74.4</td>
<td>75.8</td>
<td>74.3</td>
<td>75.1</td>
<td>76.0</td>
<td>77.2</td>
<td>78.4</td>
</tr>
<tr>
<td>recall</td>
<td>89.6</td>
<td>88.2</td>
<td>88.6</td>
<td>88.1</td>
<td>88.3</td>
<td>88.7</td>
<td>89.1</td>
<td>89.8</td>
</tr>
<tr>
<td>precision</td>
<td>69.4</td>
<td>64.3</td>
<td>66.3</td>
<td>64.2</td>
<td>65.3</td>
<td>66.7</td>
<td>68.1</td>
<td>69.6</td>
</tr>
<tr>
<td>w/o OD</td>
<td>( \sigma = 5 )</td>
<td>( \sigma = 30 )</td>
<td>w/o VS</td>
<td>K = 1</td>
<td>K = 2</td>
<td>K = 4</td>
<td>( \mu = 3 )</td>
<td>( \mu = 11 )</td>
</tr>
<tr>
<td>F-score</td>
<td>74.8</td>
<td>76.4</td>
<td>77.9</td>
<td>77.5</td>
<td>72.1</td>
<td>74.9</td>
<td>78.0</td>
<td>77.3</td>
</tr>
<tr>
<td>recall</td>
<td>88.5</td>
<td>88.7</td>
<td>89.4</td>
<td>89.2</td>
<td>85.2</td>
<td>87.3</td>
<td>89.8</td>
<td>89.6</td>
</tr>
<tr>
<td>precision</td>
<td>64.8</td>
<td>67.0</td>
<td>69.0</td>
<td>68.5</td>
<td>62.5</td>
<td>68.9</td>
<td>65.6</td>
<td>69.5</td>
</tr>
</tbody>
</table>

We assess the effectiveness of the adaptive location selection, the cross pyramid architecture, the number of the selected locations, the occlusion detection, the threshold for occlusion detection, the view selection, the levels of the pyramid architecture, and the patch size in PDE on the Ignatius dataset from training dataset provided by the Tanks and Temples benchmark. We use the complete ADEP-MVSNet with number of selected location \( S = 20,000 \), occlusion detection threshold \( \sigma = 10 \times depth \text{ interval} \), levels of the pyramid architecture \( K = 3 \), and patch size \( \mu = 7 \) as the baseline. Table 3 represents the recall, precision and F-score of the ADEP-MVSNet with the baseline settings, with uniformly random selected locations instead of adaptive selected locations (RS), with single pyramid architecture instead of cross pyramid architecture (PA), with the increasing number of selected locations (the depth map is only upsampled without adaptive depth estimation when the number of selected locations \( S = 0 \)), without the occlusion detection (w/o OD), with increasing occlusion detection threshold \( \sigma \), with increasing levels of the cross pyramid architecture, and with the increasing patch size \( \mu \).

4.5. Ablation study

We assess the effectiveness of the adaptive location selection, the cross pyramid architecture, the number of the selected locations, the occlusion detection, the threshold for occlusion detection, the view selection, the levels of the pyramid architecture, and the patch size in PDE on the Ignatius dataset from training dataset provided by the Tanks and Temples benchmark. We use the complete ADEP-MVSNet with number of selected location \( S = 20,000 \), occlusion detection threshold \( \sigma = 10 \times depth \text{ interval} \), levels of the pyramid architecture \( K = 3 \), and patch size \( \mu = 7 \) as the baseline. Table 3 represents the recall, precision and F-score of the ADEP-MVSNet with the baseline settings, with uniformly random selected locations instead of adaptive selected locations (RS), with single pyramid architecture instead of cross pyramid architecture (PA), with the increasing number of selected locations (the depth map is only upsampled without adaptive depth estimation when the number of selected locations \( S = 0 \)), without the occlusion detection (w/o OD), with increasing occlusion detection threshold \( \sigma \), with increasing levels of the cross pyramid architecture, and with the increasing patch size \( \mu \).

It can be observed that using randomly selected locations only slightly improves the performance compared to simply upsampling. The reason is that the depth estimation is mainly performed on the locations where depth hypotheses are generally accurate instead of the locations requiring further refinement. Using the single pyramid architecture also improves the performance, but compared to the cross pyramid architecture the improvement is not that significant. With an increasing number of the selected locations, the F-score first increases and then decreases after \( S = 30,000 \) as the precision decreases. The reason is that some well-estimated locations are re-estimated with erroneous depth values since the perceptive field of pixelwise estimation is relatively local compared to the method using CNN-based regularization. Without the occlusion detection, the F-score, accuracy, and completeness all decrease as the location selection is disturbed. However, compared to random location selection, the scores are a little better. A larger or smaller \( \sigma \) will lead to worse performance, as a small \( \sigma \) makes the inaccurate depth hypotheses to be refined wrongly judged as occlusion and a large \( \sigma \) makes the criteria to be defined as occlusion too strict. Wiping out the view selection will lead to both recall and precision falling, as the related views contribute equally to the cost vector construction while the conditions of different views vary a lot. With increasing levels of the cross pyramid architecture, the F-score will first increase while \( K \leq 3 \) and then decrease while \( K = 4 \). When \( K = 3 \) the resolution of the output depth map is equal to the input image. The levels fewer than 3 cannot make full use of the input information and only output downscaled depth maps, while more levels will not introduce extra information for geometry estimation and may induce more outliers. Increasing the patch size will increase the completeness but decrease the precision, as the larger patch will contain more contextual information while neglecting the local information.

4.6. Limitations

Our method can not correctly estimate depth values for some tiny or thin objects which are ignored in the initial depth maps. To be specific, if the tiny or thin objects are falsely estimated in all views participating in the reference depth estimation in the initialization stage, they will not be recovered in the subsequent stages as none of the correct depth candidates near the true depth values can be constructed. Fig. 10 demonstrates two examples where our method fails to estimate the correct depth values for thin objects.
5. Conclusion and future work

We propose a Multi-View Stereo (MVS) network which can perform efficient high-resolution depth estimation while the growth of the required time and memory is only logarithmic with output resolution. We first introduce the cross pyramid architecture. Compared to the pyramid architecture that gradually refines the depth map with only feature information from related views, the cross pyramid architecture further utilizes the estimated depth hypotheses from related views for depth estimation in the reference view. By introducing the location selection strategy, our depth estimation module can concentrate on locations where depth values are falsely estimated and avoid unnecessary computation on well-estimated regions. The proposed pixelwise depth estimation module is lightweight and can effectively perform pixelwise depth estimation on selected locations. Experiments on the Tanks and Temples benchmark demonstrate that our method can reconstruct high-quality and high-resolution models with less GPU memory consumption. In the future, we are going to encode the reprojection error values as geometric consistency information into the cost construction procedure. The geometric consistency can be utilized to constrain depth hypotheses across multiple views to be consistent in 3D space and plays a critical role in depth map fusion. It has been widely considered in various traditional MVS methods and validated to be effective, but is still not introduced to learning-based MVS. Besides, inspired by the AANet proposed by Xu et al. [33], we will consider to apply adaptive sampling instead of regular sampling for the feature aggregation procedure in the pixelwise depth estimation. The adaptive sampling strategy can fit the structure of the captured objects and alleviate the well-known edge-flattening issue.

Declaration of Competing Interest

We wish to confirm that there are no known conflicts of interest associated with this publication and there has been no significant financial support for this work that could have influenced its outcome. We confirm that the manuscript has been read and approved by all named authors and that there are no other persons who satisfied the criteria for authorship but are not listed. We further confirm that the order of authors listed in the manuscript has been approved by all of us. We confirm that we have given due consideration to the protection of intellectual property associated with this work and that there are no impediments to publication, including the timing of publication, with respect to intellectual property. In so doing we confirm that we have followed the regulations of our institutions concerning intellectual property. We understand that the Corresponding Author is the sole contact for the Editorial process (including Editorial Manager and direct communications with the office). He/she is responsible for communicating with the other authors about progress, submissions of revisions and final approval of proofs. We confirm that we have provided a current, correct email address which is accessible by the Corresponding Author.

CRediT authorship contribution statement

Jie Liao: Conceptualization, Methodology, Software, Validation, Writing - original draft. Yanping Fu: Data curation, Resources, Visualization, Software. Qingnan Yan: Data curation, Resources, Visualization. Fei Luo: Writing - review & editing, Supervision. Chunxia Xiao: Writing - review & editing, Supervision.

Acknowledgment

This work is partially supported by the Key Technological Innovation Projects of Hubei Province (2018AAA062), NSFC (NO. 61972298), and Wuhan University-Huawei Geoinformatics Innovation Lab.

References


