Point-Based Multi-View Stereo Network

Rui Chen[1,3*] Songfang Han[2,3*] Jing Xu[1] Hao Su[3]

[1]Tsinghua University [2]The Hong Kong University of Science and Technology

[3]University of California, San Diego

chenr17@mails.tsinghua.edu.cn shanaf@connect.ust.hk
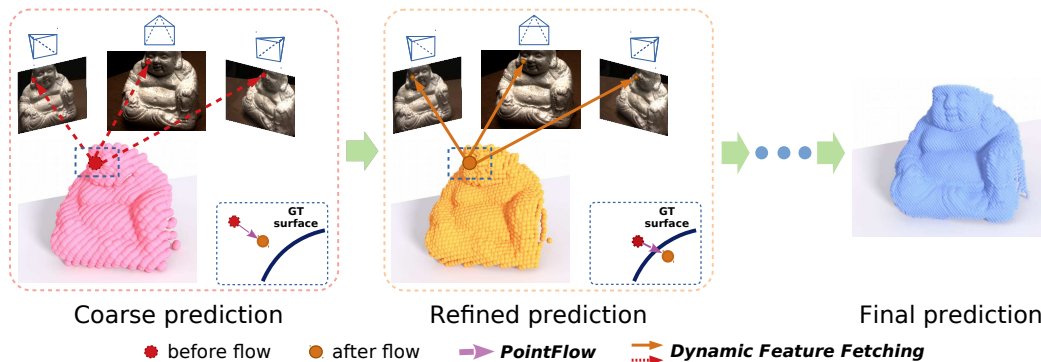
jingxu@tsinghua.edu.cn haosu@eng.ucsd.edu

Figure 1: Point-MVSNet performs multi-view stereo reconstruction in a coarse-to-fine fashion, learning to predict the 3D flow of each point to the groundtruth surface based on geometry priors and 2D image appearance cues dynamically fetched from multi-view images and regress accurate and dense point clouds iteratively.

Abstract

We introduce Point-MVSNet, a novel point-based deep framework for multi-view stereo (MVS). Distinct from existing cost volume approaches, our method directly processes the target scene as point clouds. More specifically, our method predicts the depth in a coarse-to-fine manner. We first generate a coarse depth map, convert it into a point cloud and refine the point cloud iteratively by estimating the residual between the depth of the current iteration and that of the ground truth. Our network leverages 3D geometry priors and 2D texture information jointly and effectively by fusing them into a feature-augmented point cloud, and processes the point cloud to estimate the 3D flow for each point. This point-based architecture allows higher accuracy, more computational efficiency and more flexibility than cost-volume-based counterparts. Experimental results show that our approach achieves a significant improvement in reconstruction quality compared with state-of-the-art methods on the DTU and the Tanks and Temples dataset. Our source code and trained models are available at https://github.com/callmeray/PointMVSNet.

1. Introduction

Recent learning-based multi-view stereo (MVS) methods [12, 29, 10] have shown great success compared with their traditional counterparts as learning-based approaches are able to learn to take advantage of scene global semantic information, including object materials, specularity, and environmental illumination, to get more robust matching and more complete reconstruction. All these approaches apply dense multi-scale 3D CNNs to predict the depth map or voxel occupancy. However, 3D CNNs require memory cubic to the model resolution, which can be potentially prohibitive to achieving optimal performance. While Maxim et al. [24] addressed this problem by progressively generating an Octree structure, the quantization artifacts brought by grid partitioning still remain, and errors may accumulate since the tree is generated layer by layer.

In this work, we propose a novel point cloud multi-view stereo network, where the target scene is directly processed as a point cloud, a more efficient representation, particularly when the 3D resolution is high. Our framework is composed of two steps: first, in order to carve out the approximate object surface from the whole scene, an initial coarse depth map is generated by a relatively small 3D cost volume and

then converted to a point cloud. Subsequently, our novel *PointFlow* module is applied to iteratively regress accurate and dense point clouds from the initial point cloud. Similar to ResNet [8], we explicitly formulate the *PointFlow* to predict the residual between the depth of the current iteration and that of the ground truth. The 3D flow is estimated based on geometry priors inferred from the predicted point cloud and the 2D image appearance cues dynamically fetched from multi-view input images (Figure 1).

We find that our Point-based Multi-view Stereo Network (Point-MVSNet) framework enjoys advantages in accuracy, efficiency, and flexibility when it is compared with previous MVS methods that are built upon a predefined 3D volume with the fixed resolution to aggregate information from views. Our method adaptively samples potential surface points in the 3D space. It keeps the continuity of the surface structure naturally, which is necessary for high precision reconstruction. Furthermore, because our network only processes valid information near the object surface instead of the whole 3D space as is the case in 3D CNNs, the computation is much more efficient. Lastly, the adaptive refinement scheme allows us to first peek at the scene at coarse resolution and then densify the reconstructed point cloud only in the region of interest. For scenarios such as interaction-oriented robot vision, this flexibility would result in saving of computational power.

Our method achieves state-of-the-art performance on standard multi-view stereo benchmarks among learning-based methods, including DTU [1] and Tanks and Temples [15]. Compared with previous state-of-the-art, our method produces better results in terms of both completeness and overall quality. Besides, we show potential applications of our proposed method, such as foveated depth inference.

2. Related work

Multi-view Stereo Reconstruction   MVS is a classical problem that had been extensively studied before the rise of deep learning. A number of 3D representations are adopted, including volumes [26, 9], deformation models [3, 31], and patches [5], which are iteratively updated through multi-view photometric consistency and regularization optimization. Our iterative refinement procedure shares a similar idea with these classical solutions by updating the depth map iteratively. However, our learning-based algorithm achieves improved robustness to input image corruption and avoids the tedious manual hyper-parameters tuning.

Learning-based MVS   Inspired by the recent success of deep learning in image recognition tasks, researchers began to apply learning techniques to stereo reconstruction tasks for better patch representation and matching [7, 22, 16]. Although these methods in which only 2D networks are used have made a great improvement on stereo tasks, it is

difficult to extend them to multi-view stereo tasks, and their performance is limited in challenging scenes due to the lack of contextual geometry knowledge. Concurrently, 3D cost volume regularization approaches have been proposed [14, 12, 13], where a 3D cost volume is built either in the camera frustum or the scene. Next, the 2D image features of multi-views are warped in the cost volume, so that 3D CNNs can be applied to it. The key advantage of 3D cost volume is that the 3D geometry of the scene can be captured by the network explicitly, and the photometric matching can be performed in 3D space, alleviating the influence of image distortion caused by perspective transformation and potential occlusions, which makes these methods achieve better results than 2D learning based methods. Instead of using voxel grids, in this paper we propose to use a point-based network for MVS tasks to take advantage of 3D geometry learning without being buredened by the inefficiency found in 3D CNN computation.

High-Resolution MVS   High-resolution MVS is critical to real applications such as robot manipulation and augmented reality. Traditional methods [17, 5, 18] generate dense 3D patches by expanding from confident matching key-points repeatedly, which is potentially time-consuming. These methods are also sensitive to noise and change of viewpoint owing to the usage of hand-crafted features. Recent learning methods try to ease memory consumption by advanced space partitioning [21, 27, 24]. However, most of these methods construct a fixed cost volume representation for the whole scene, lacking flexibility. In our work, we use point clouds as the representation of the scene, which is more flexible and enables us to approach the accurate position progressively.

Point-based 3D Learning   Recently, a new type of deep network architecture has been proposed in [19, 20], which is able to process point clouds directly without converting them to volumetric grids. Compared with voxel-based methods, this kind of architecture concentrates on the point cloud data and saves unnecessary computation. Also, the continuity of space is preserved during the process. While PointNets have shown significant performance and efficiency improvement in various 3D understanding tasks, such as object classification and detection [20], it is under exploration how this architecture can be used for MVS task, where the 3D scene is unknown to the network. In this paper, we propose *PointFlow* module, which estimates the 3D flow based on joint 2D-3D features of *point hypotheses*.

3. Method

This section describes the detailed network architecture of Point-MVSNet (Figure 2). Our method can be divided into two steps, coarse depth prediction, and iterative depth
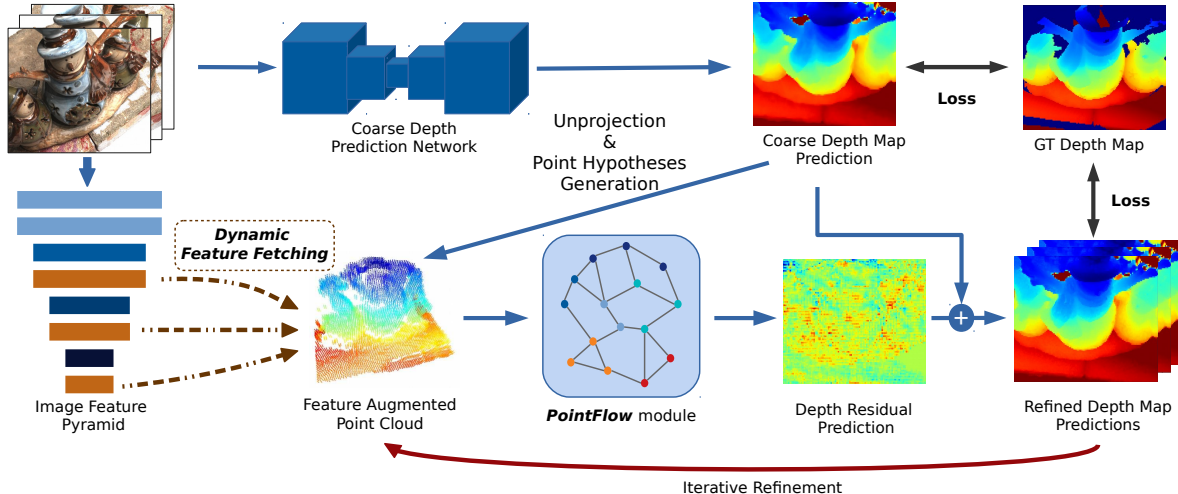
Figure 2: Overview of Point-MVSNet architecture. A coarse depth map is first predicted with low GPU memory and computation cost and then unprojected to a point cloud along with hypothesized points. For each point, the feature is fetched from the multi-view image feature pyramid dynamically. The *PointFlow* module uses the feature augmented point cloud for depth residual prediction, and the depth map is refined iteratively.

refinement. Let $\mathbf{I}_0$ denote the reference image and $\{\mathbf{I}_i\}_{i=1}^N$ denote a set of its neighbouring source images. We first generate a coarse depth map for $\mathbf{I}_0$ (Section 3.1). Since the resolution is low, the existing volumetric MVS method has sufficient efficiency and can be used. Second we introduce the 2D-3D feature lifting (Section 3.2), which associates the 2D image information with 3D geometry priors. Then we propose our novel *PointFlow* module (Section 3.3) to iteratively refine the input depth map to higher resolution with improved accuracy.

3.1. Coarse depth prediction

Recently, learning-based MVS [12, 29, 11] achieves state-of-the-art performance using multi-scale 3D CNNs on cost volume regularization. However, this step could be extremely memory expensive as the memory requirement is increasing cubically as the cost volume resolution grows. Taking memory and time into consideration, we use the recently proposed MVSNet [29] to predict a relatively low-resolution cost volume.

Given the images and corresponding camera parameters, MVSNet [29] builds a 3D cost volume upon the reference camera frustum. Then the initial depth map for reference view is regressed through multi-scale 3D CNNs and the soft argmin [15] operation. In MVSNet, feature maps are downsampled to $1/4$ of the original input image in each dimension and the number of virtual depth planes are 256 for both training and evaluation. On the other hand, in our coarse depth estimation network, the cost volume is constructed with feature maps of $1/8$ the size of the reference image, containing 48 or 96 virtual depth planes for training and evaluation, respectively. Therefore, our memory usage of this 3D feature volume is about $1/20$ of that in MVSNet.

3.2. 2D-3D feature lifting

Image Feature Pyramid Learning-based image features are demonstrated to be critical to boosting up dense pixel correspondence quality [29, 23]. In order to endow points with a larger receptive field of contextual information at multiple scales, we construct a 3-scale feature pyramid. 2D convolutional networks with stride 2 are applied to downsample the feature map, and each last layer before the downsampling is extracted to construct the final feature pyramid $\mathbf{F}_i = [\mathbf{F}_i^1, \mathbf{F}_i^2, \mathbf{F}_i^3]$ for image $\mathbf{I}_i$. Similar to common MVS methods[29, 11], feature pyramids are shared among all input images.

Dynamic Feature Fetching The point feature used in our network is compromised of the fetched multi-view image feature variance with the normalized 3D coordinates in world space $\mathbf{X}_p$. We will introduce them separately.

Image appearance features for each 3D point can be fetched from the multi-view feature maps using a differentiable unprojection given corresponding camera parameters. Note that features $\mathbf{F}_i^1, \mathbf{F}_i^2, \mathbf{F}_i^3$ are at different image resolutions, thus the camera intrinsic matrix should be scaled at each level of the feature maps for correct feature warping. Similar to MVSNet [29], we keep a variance-based cost metric, i.e. the feature variance among different views, to aggregate features warped from an arbitrary number of views. For pyramid feature at level j, the variance metric for N views is defined as below:

$$\mathbf{C}^j = \frac{\sum_{i=1}^N \left(\mathbf{F}_i^j - \overline{\mathbf{F}^j}\right)^2}{N}, (j = 1, 2, 3). \qquad (1)$$

To form the features residing at each 3D point, we do a concatenation of the fetched image feature and the normalized point coordinates:

$$\mathbf{C}_p = \text{concat}[\mathbf{C}_p^j, \mathbf{X}_p], (j = 1, 2, 3). \qquad (2)$$

This feature augmented point $\mathbf{C}_p$ is the input to our *PointFlow* module.

As shall be seen in the next section, since we are predicting the depth residual iteratively, we update the point position $\mathbf{X}_p$ after each iteration and fetch the point feature $\mathbf{C}_p^k$ from the image feature pyramid, an operation we name as *dynamic feature fetching*. Note that this step is distinct from cost-volume-based methods, by which the fetched features at each voxel are determined by the fixed space partition of the scene. In contrast, our method can fetch features from different areas of images dynamically according to the updated point position. Therefore, we can concentrate on the regions of interest in the feature maps, instead of treating them uniformly.

3.3. *PointFlow*

Depth maps generated from Section 3.1 have limited accuracy due to the low spatial resolution of 3D cost volume. We propose *PointFlow*, our novel approach to iteratively refine the depth map.

With known camera parameters, we first un-project the depth map to be a 3D point cloud. For each point, we aim to estimate its displacement to the ground truth surface along the reference camera direction by observing its neighboring points from all views, so as to push the points to *flow* to the target surface. Next, we discuss the components of our module in detail.

Point Hypotheses Generation It is non-trivial to regress depth displacement of each point from the extracted image feature maps. Due to perspective transformation, the spatial context embedded in 2D feature maps cannot reflect the proximity in 3D Euclidean space.

In order to facilitate the modeling of network, we propose to generate a sequence of *point hypotheses* $\tilde{\mathbf{p}}$ with different displacement along the reference camera direction as shown in Figure 3. Let $\mathbf{t}$ denote the normalized reference camera direction, and s denote the displacement step size. For an unprojected point $\mathbf{p}$, its hypothesized point set $\{\tilde{\mathbf{p}}_k\}$ is generated by

$$\tilde{\mathbf{p}}_k = \mathbf{p} + ks\mathbf{t}, \quad k = -m, \ldots, m. \qquad (3)$$

These *point hypotheses* are critical for the network to infer the displacement, for the necessary neighbourhood image feature information at different depth are gathered in these points along with spatial geometry relationship.
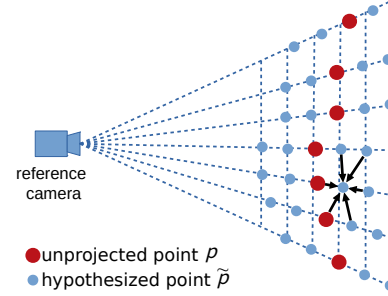

Figure 3: Illustraion of point hypotheses generation and edge construction: For each unprojected point p, the $2m + 1$ point hypotheses $\{\tilde{\mathbf{p}}_k\}$ are generated along the reference camera direction. Directed edges are constructed between each hypothesized point and its *kNN* points for edge convolution.

Edge Convolution Classical MVS methods have demonstrated that local neighborhood is important for robust depth prediction. Similarly, we take the strategy of recent work DGCNN [28] to enrich feature aggregation between neighboring points. As shown in Figure 3, a directed graph is constructed on the point set using k nearest neighbors (*kNN*), such that local geometric structure information could be used for the feature propagation of points.

Denote the feature augmented point cloud by $\mathbf{C}_{\tilde{p}} = \{\mathbf{C}_{\tilde{p}_1}, \ldots, \mathbf{C}_{\tilde{p}_n}\}$, then edge convolution is defined as:

$$\mathbf{C}'_{\tilde{p}} = \underset{q \in kNN(\tilde{p})}{\square} h_\Theta \left(\mathbf{C}_{\tilde{p}}, \ \mathbf{C}_{\tilde{p}} - \mathbf{C}_q\right), \qquad (4)$$

where h_Θ is a learnable non-linear function parameterized by Θ, and $\square$ is a channel-wise symmetric aggregation operation. There are multiple options for the symmetry operation, including max pooling, average pooling, and weighted sum. We compared max pooling and average pooling and observed similar performance after tuning hyper-parameters carefully.

Flow Prediction The network architecture for flow prediction is shown in Figure 4. The input is a feature augmented point cloud, and the output is a depth residual map. We use three EdgeConv layers to aggregate point features at different scales of the neighborhood. Shortcut connections are used to include all the EdgeConv outputs as local point features. Finally, a shared multilayer perceptron (MLP) is used to transform the pointwise features, which outputs a probability scalar with softmax among hypothesized points of each unprojected point. The displacement of the unprojected points are predicted as the probabilistic weighted sum of the displacement among all predicted point hypotheses:

$$\Delta d_p = \mathbf{E}(ks) = \sum_{k=-m}^{m} ks \times \text{Prob}(\tilde{\mathbf{p}}_k). \qquad (5)$$
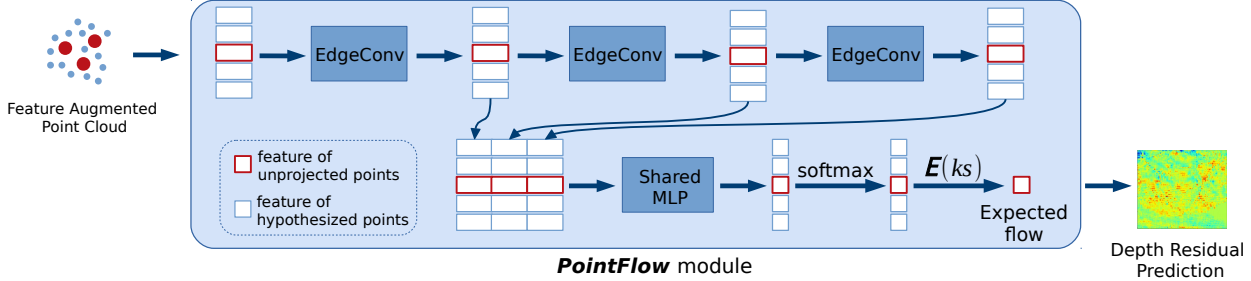
Figure 4: Network architecture of the proposed *PointFlow* module.

Note that this operation is differentiable. The output depth residual map is obtained by projecting the displacement back, which will be added to the initial input depth map for depth refinement.

Iterative Refinement with Upsampling Because of the flexibility of our point-based network architecture, the flow prediction can be performed iteratively, which is much harder for 3D cost-volume-based methods, because the space partitioning is fixed after the construction of cost volume. For depth map $\mathbf{D}^{(i)}$ from coarse prediction or former residual prediction, we can first upsample it using nearest neighbor to higher spatial resolution and then perform the flow prediction to obtain $\mathbf{D}^{(i+1)}$. Moreover, we decrease the depth interval s between the unprojected points and hypothesized points at each iteration, so that more accurate displacement can be predicted by capturing more detailed features from closer point hypotheses.

3.4. Training loss

Similar to most deep MVS networks, we treat this problem as a regression task and train the network with the L_1 loss, which measures the absolute difference between the predicted depth map and the groundtruth depth map. Losses for the initial depth map and iteratively refined ones are all considered:

$$Loss = \sum_{i=0}^{l} \left(\frac{\lambda^{(i)}}{s^{(i)}} \sum_{p \in \mathbf{P}_{\text{valid}}} \left\| \mathbf{D}_{\text{GT}}(p) - \mathbf{D}^{(i)}(p) \right\|_1 \right),$$
(6)

where $\mathbf{P}_{\text{valid}}$ represents the valid groundtruth pixel set and l is the iteration number. The weight $\lambda^{(i)}$ is set to 1.0 in training.

4. Experiments

4.1. DTU dataset

The DTU dataset [1] is a large-scale MVS dataset, which consists of 124 different scenes scanned in 7 different lighting conditions at 49 or 64 positions. The data for each scan is composed of an RGB image and corresponding

	Acc. (mm)	Comp. (mm)	Overall (mm)
Camp [2]	0.835	0.554	0.695
Furu [5]	0.613	0.941	0.777
Tola [25]	0.342	1.190	0.766
Gipuma [6]	**0.283**	0.873	0.578
SurfaceNet [12]	0.450	1.040	0.745
MVSNet [29]	0.396	0.527	0.462
Ours	0.361	0.421	0.391
Ours-HiRes	0.342	**0.411**	**0.376**

Table 1: Quantitative results of reconstruction quality on the DTU evaluation dataset (lower is better).

camera parameters. The dataset is split into training, validation, and evaluation sets.

4.2. Implementation details

Training We train Point-MVSNet on the DTU training dataset. For data pre-processing, we follow MVSNet [29] to generate depth maps from the given groundtruth point clouds. During training, we set input image resolution to 640×512, and number of views to $N = 3$. The input view set is chosen with the same view selection strategy as in MVSNet (Section 4.1). For coarse prediction, we construct a 3D cost volume with $D = 48$ virtual depth planes, which are uniformly sampled from 425mm to 921mm. For the depth refinement step, we set flow iterations $l = 2$, with depth intervals being 8mm and 4mm, respectively. The number of nearest neighbor points $= 16$. We use RMSProp of initial learning rate 0.0005 which is decreased by 0.9 for every 2 epochs. The coarse prediction network is trained alone for 4 epochs, and then, the model is trained end-to-end for another 12 epochs. Batch size is set to 4 on 4 NVIDIA GTX 1080Ti graphics cards.

Evaluation We use $D = 96$ depth layers for initial depth prediction and set flow iterations $l = 3$ for depth refinement. We predict the reference view depth map for each $N = 5$ view set. Then we fuse all depth maps to point clouds using same post-processing provided by [29]. We evaluate our method in two different input image resolutions: 1280×960 ("Ours"), and 1600×1152 ("Ours-HiRes").
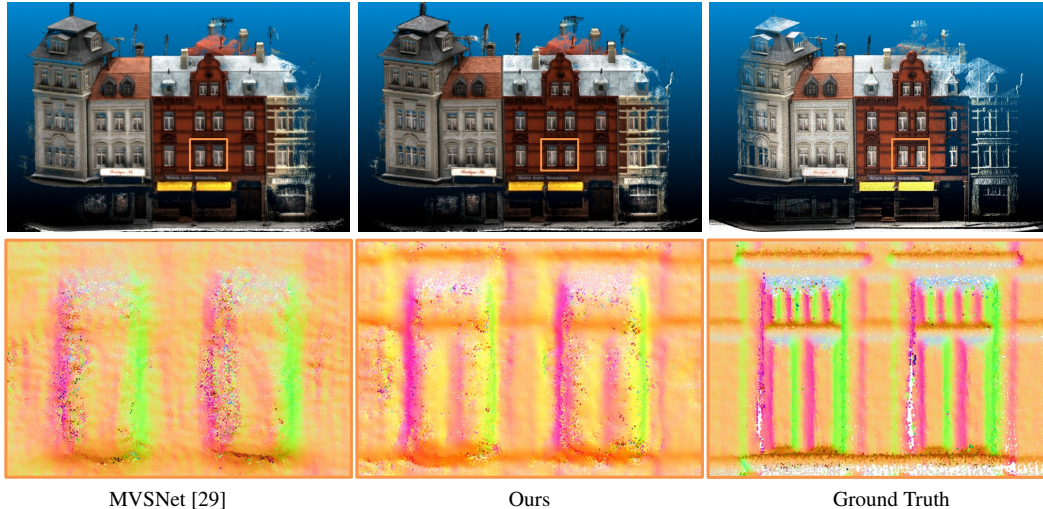
| MVSNet [29] | Ours | Ground Truth |

Figure 5: Qualitative results of *scan* 9 of DTU dataset. Top: Whole point cloud. Bottom: Visualization of normals in zoomed local area. Our Point-MVSNet generates detailed point clouds with more high-frequency component than MVSNet. For fair comparison, the depth maps predicted by MVSNet are interpolated to the same resolution as our method.
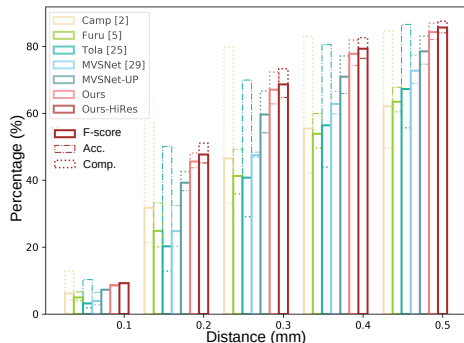

Figure 6: F-score, accuracy and completeness of different distance thresholds on the DTU evaluation dataset (higher is better). For fair comparison, we upsample the depth map predicted by MVSNet to the same resolution as our method before depth fusion (288×216 to 640×480).

4.3. Benchmarking on DTU dataset

We evaluate the proposed method on the DTU evaluation dataset. Quantitative results are shown in Table 1 and Figure 6, where the accuracy and completeness are computed using the official code from the DTU dataset, and the *f-score* is calculated as mentioned in [15] as the measure of overall performance of accuracy and completeness. While Gipuma [6] performs the best in terms of accuracy, our Point-MVSNet outperforms start-of-the-art in both completeness and overall quality. Qualitative results are shown in Figure 5. Point-MVSNet generates a more detailed point cloud compared with MVSNet. Especially in those edgy areas, our method can capture high-frequency geometric features.

4.4. *PointFlow* iteration

Because of the continuity and flexibility of point representation, the refinement and densification can be performed iteratively on former predictions to give denser and more accurate predictions. While the model is trained using $l = 2$ iterations, we test the model using iteration ranging from 0 to 3. For each iteration, we upsample the point cloud and decrease the depth interval of point hypotheses simultaneously, enabling the network to capture more detailed features. We compare the reconstruction quality, depth map resolution, GPU memory consumption and runtime at different iterations, along with performance reported by state-of-the-art methods in Table 2. The reconstruction quality improves significantly with multiple iterations, which verifies the effectiveness of our methods. Note that our method already outperforms the state-of-the-art after the second iteration. Qualitative results are shown in Figure 7.

4.5. Ablation study

In this section we provide ablation experiments and quantitative analysis to evaluate the strengths and limitations of the key components in our framework. For all the following studies, experiments are performed and evaluated on the DTU dataset, and both *accuracy* and *completeness* are used to measure the reconstruction quality. We set the iteration number to $l = 2$, and all other experiment settings are the same as Section 4.3.

Edge Convolution By replacing the edge convolution in Equation (4) with geometry-unaware feature aggregation:

$$\mathbf{C}'_{\tilde{p}} = \underset{q \in kNN(\tilde{p})}{\square} h_{\Theta}\left(\mathbf{C}_q\right), \tag{7}$$
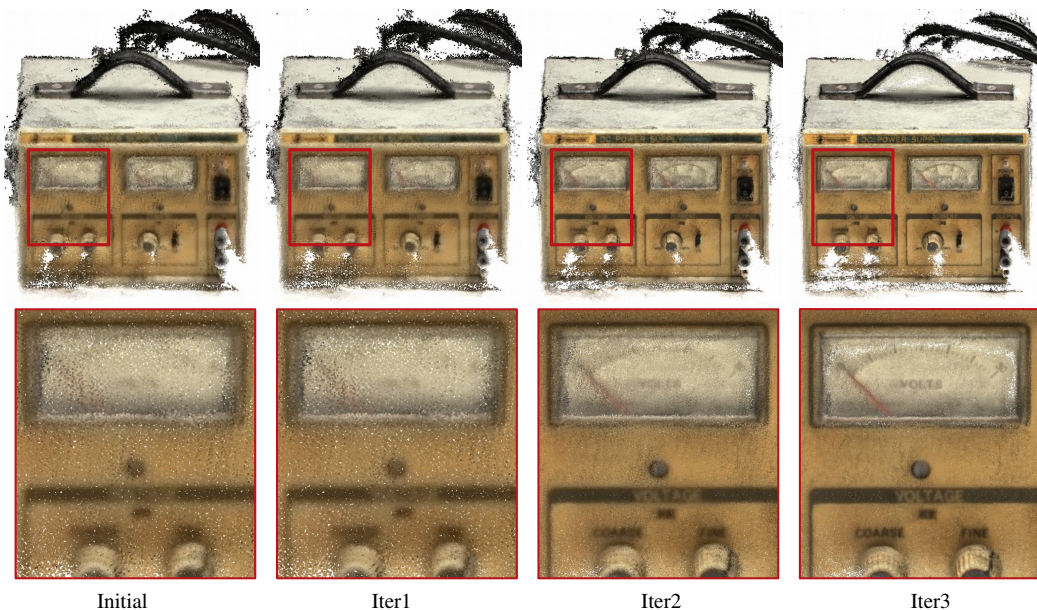
Figure 7: Qualitative results at different flow iterations. Top: Whole point cloud. Bottom: Zoomed local area. The generated point cloud becomes denser after each iteration, and more geometry details can be captured.

Iter.	Acc. (mm)	Comp. (mm)	Overall (mm)	0.5mm *f-score*	Depth Map Res.	Depth Interval (mm)	GPU Mem. (MB)	Runtime (s)
-	0.693	0.758	0.726	47.95	160×120	5.30	**7219**	**0.34**
1	0.674	0.750	0.712	48.63	160×120	5.30	7221	0.61
2	0.448	0.487	0.468	76.08	320×240	4.00	7235	1.14
3	**0.361**	**0.421**	**0.391**	**84.27**	**640×480**	**0.80**	8731	3.35
MVSNet[29]	0.456	0.646	0.551	71.60	288×216	2.65	10805	1.05

Table 2: Comparison result at different flow iterations measured by reconstruction quality and depth map resolution on the DTU evaluation set. Due to the GPU memory limitation, we decrease the resolution of MVSNet [29] to 1152×864×192.

where the features of neighbor points are treated equally with no regard for their geometric relationship to the centroid point, the reconstruction quality drops significantly as shown in Table 3, which illustrates the importance of local neighborhood relationship information (captured by $\mathbf{C}_{\tilde{p}} - \mathbf{C}_q$) for feature aggregation.

Euclidean Nearest Neighbour   In this part, we construct the directed graph $\mathcal{G}$ using points belonging to adjacent pixels in the reference image, instead of searching the k-NN points, which leads to decreased reconstruction quality. The reason is that, for images of 3D scenes, near-by pixels may correspond to distant objects due to occlusion. Therefore, using neighboring points in the image space may aggregate irrelevant features for depth residual prediction, leading to descending performance.

Feature Pyramid   In this part, point cloud only fetches features from the last layer of the feature map, instead of from the whole feature pyramid. As shown in Table 3, in contrast to the relatively stable performance for changing edge convolution strategies as discussed above, the drop will

be significant in the absence of the other two components, which demonstrates the effectiveness of the leveraging context information at different scales for feature fetching.

4.6. Reliance on initial depth maps

Our method uses state-of-the-art approaches to get a coarse depth map prediction, which is then iteratively refined by predicting depth residuals. We found that our approach is robust to noisy initial depth estimation in a certain range through the following experiments. We added Gaussian noise of different scales to the initial depth map and evaluated the reconstruction error. Figure 8 shows that the error increases slowly and is smaller than MVSNet within 6mm noise.

4.7. Comparison to point cloud upsampling

Our work can also be considered as a data-driven point cloud upsampling method with assisting information from reference views. Therefore, we compare our method with PU-Net [30], where multi-level features are extracted from the coarse point cloud to reconstruct an upsampled point cloud.

EDGE	EUCNN	PYR	Acc. (mm)	Comp. (mm)
✓	✓	✓	**0.448**	**0.487**
✓	✓	✗	0.455	0.489
✓	✗	✓	0.455	0.492
✗	✓	✓	0.501	0.518
✓	✗	✗	0.475	0.504
✗	✓	✗	0.574	0.565
✗	✗	✓	0.529	0.532

Table 3: Ablation study on network architectures on the DTU evaluation dataset, which demonstrates the effectiveness of different components. EDGE denotes edge convolution, EUCNN denotes grouping by nearest neighbour points in Euclidean distance, and PYR denotes the usage of image feature pyramid.
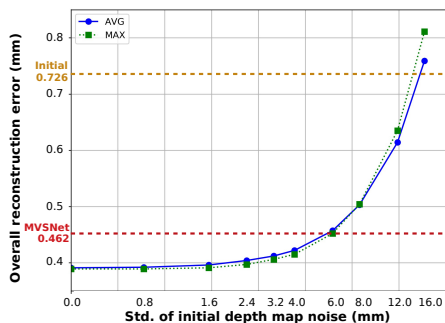

Figure 8: Reconstruction error w.r.t. initial depth map noise. AVG denotes average pooling, MAX denotes max pooling.

	Acc. (mm)	Comp. (mm)	Overall (mm)
PU-Net [30]	1.220	0.667	0.943
Ours	**0.361**	**0.421**	**0.391**

Table 4: Comparison of reconstruction quality on the DTU evaluation dataset with PU-Net [30].

We use the same coarse depth prediction network as in our model, and train PU-Net to upsample the coarse point cloud. We use the same joint loss as mentioned in their paper, which consists of two losses — the Earth Mover's distance (EMD) [4] loss between the predicted point cloud and the reference groundtruth point cloud and a repulsion loss. For evaluation, the PU-Net is applied on the coarse predicted point cloud twice to generate a denser point cloud with 16 times more points. Quantitative result is shown in Table 4. Our Point-MVSNet can generate a more accurate point cloud from the coarse one by inducing *flow* for each point from observation of context information in multi-view images.

4.8. Foveated depth inference

The point-based network architecture enables us to process an arbitrary number of points. Therefore, instead of upsampling and refining the whole depth map, we can choose to only infer the depth in the region of interest (ROI)

Figure 9: Illustration of foveated depth inference with our proposed method. Different point density levels are denoted by different colors: Gray for sparsest, Brown for intermediate, Green for densest.

based on the input image or the predicted coarse depth map. As shown in Figure 9, we generate a point cloud of three different density levels by only upsampling and refining the ROI in the previous stage.

4.9. Generalizability of the *PointFlow* Module

In order to evaluate the generalizability of our *PointFlow* module, we test it on the Tanks and Temples intermediate dataset [15], which is a large outdoor dataset captured in complex environments. We first generate coarse depth maps using MVSNet [29], and then apply our *PointFlow* module to refine them. The *f-score* increases from 43.48 to 48.27 (larger is better) and the rank rises from 13.12 to 7.25 (lower is better, date: Mar. 22, 2019). Reconstructed point clouds are shown in supplementary materials.

5. Conclusion

We present a novel point-based architecture for high-resolution multi-view stereo reconstruction. Instead of building a high-resolution cost volume, our proposed Point-MVSNet processes the scene as a point cloud directly, which reduces unnecessary computation and preserves the spatial continuity. Experiments show that Point-MVSNet is able to produce high-quality reconstruction point clouds on benchmarks. Additionally, Point-MVSNet is applicable to foveated depth inference to greatly reducing computation, which cannot be easily implemented for cost-volume-based methods.

Acknowledgement

References

[1] Henrik Aanæs, Rasmus RamsbØl Jensen, George Vogiatzis, Engin Tola, and Anders Bjorholm Dahl. Large-Scale Data for Multiple-View Stereopsis. *Int. J. Comput. Vision*, 120(2):153–168, Nov. 2016.

[2] Neill D. F. Campbell, George Vogiatzis, Carlos Hernández, and Roberto Cipolla. Using Multiple Hypotheses to Improve Depth-Maps for Multi-View Stereo. In David Forsyth, Philip Torr, and Andrew Zisserman, editors, *Computer Vision – ECCV 2008*, volume 5302, pages 766–779. 2008.

[3] Carlos Hernández Esteban and Francis Schmitt. Silhouette and stereo fusion for 3d object modeling. *Computer Vision and Image Understanding*, 96(3):367–392, 2004.

[4] Haoqiang Fan, Hao Su, and Leonidas J Guibas. A point set generation network for 3d object reconstruction from a single image. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 605–613, 2017.

[5] Y. Furukawa and J. Ponce. Accurate, Dense, and Robust Multiview Stereopsis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(8):1362–1376, Aug. 2010.

[6] Silvano Galliani, Katrin Lasinger, and Konrad Schindler. Gipuma: Massively parallel multi-view stereo reconstruction. *Publikationen der Deutschen Gesellschaft für Photogrammetrie, Fernerkundung und Geoinformation e. V*, 25:361–369, 2016.

[7] Xufeng Han, Thomas Leung, Yangqing Jia, Rahul Sukthankar, and Alexander C Berg. Matchnet: Unifying feature and metric learning for patch-based matching. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3279–3286, 2015.

[8] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[9] Alexander Hornung and Leif Kobbelt. Hierarchical volumetric multi-view stereo reconstruction of manifold surfaces based on dual graph embedding. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, volume 1, pages 503–510. IEEE, 2006.

[10] Po-Han Huang, Kevin Matzen, Johannes Kopf, Narendra Ahuja, and Jia-Bin Huang. DeepMVS: Learning Multi-view Stereopsis. *arXiv:1804.00650 [cs]*, Apr. 2018.

[11] Sunghoon Im, Hae-Gon Jeon, Stephen Lin, and In So Kweon. DPSNet: End-to-end Deep Plane Sweep Stereo. Sept. 2018.

[12] Mengqi Ji, Juergen Gall, Haitian Zheng, Yebin Liu, and Lu Fang. SurfaceNet: An End-to-end 3D Neural Network for Multiview Stereopsis. *arXiv:1708.01749 [cs]*, Aug. 2017.

[13] Abhishek Kar, Christian Häne, and Jitendra Malik. Learning a multi-view stereo machine. In *Advances in neural information processing systems*, pages 365–376, 2017.

[14] Alex Kendall, Hayk Martirosyan, Saumitro Dasgupta, Peter Henry, Ryan Kennedy, Abraham Bachrach, and Adam Bry. End-to-end learning of geometry and context for deep stereo regression. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 66–75, 2017.

[15] Arno Knapitsch, Jaesik Park, Qian-Yi Zhou, and Vladlen Koltun. Tanks and Temples: Benchmarking Large-scale Scene Reconstruction. *ACM Trans. Graph.*, 36(4):78:1–78:13, July 2017.

[16] Patrick Knobelreiter, Christian Reinbacher, Alexander Shekhovtsov, and Thomas Pock. End-to-end training of hybrid cnn-crf models for stereo. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2339–2348, 2017.

[17] Maxime Lhuillier and Long Quan. A quasi-dense approach to surface reconstruction from uncalibrated images. *IEEE transactions on pattern analysis and machine intelligence*, 27(3):418–433, 2005.

[18] A. Owens, J. Xiao, A. Torralba, and W. Freeman. Shape Anchors for Data-Driven Multi-view Reconstruction. In *2013 IEEE International Conference on Computer Vision*, pages 33–40, Dec. 2013.

[19] Charles R. Qi, Hao Su, Kaichun Mo, and Leonidas J. Guibas. PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation. *arXiv:1612.00593 [cs]*, Dec. 2016.

[20] Charles R. Qi, Li Yi, Hao Su, and Leonidas J. Guibas. PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space. *arXiv:1706.02413 [cs]*, June 2017.

[21] Gernot Riegler, Ali Osman Ulusoy, and Andreas Geiger. Octnet: Learning deep 3d representations at high resolutions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3577–3586, 2017.

[22] Akihito Seki and Marc Pollefeys. Sgm-nets: Semi-global matching with neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 231–240, 2017.

[23] Chengzhou Tang and Ping Tan. BA-Net: Dense Bundle Adjustment Network. June 2018.

[24] Maxim Tatarchenko, Alexey Dosovitskiy, and Thomas Brox. Octree generating networks: Efficient convolutional architectures for high-resolution 3d outputs. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2088–2096, 2017.

[25] Engin Tola, Christoph Strecha, and Pascal Fua. Efficient Large-scale Multi-view Stereo for Ultra High-resolution Image Sets. *Mach. Vision Appl.*, 23(5):903–920, Sept. 2012.

[26] George Vogiatzis, Carlos Hernández Esteban, Philip HS Torr, and Roberto Cipolla. Multiview stereo via volumetric graph-cuts and occlusion robust photo-consistency. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(12):2241–2246, 2007.

[27] Peng-Shuai Wang, Yang Liu, Yu-Xiao Guo, Chun-Yu Sun, and Xin Tong. O-CNN: Octree-based Convolutional Neural Networks for 3D Shape Analysis. *ACM Transactions on Graphics (SIGGRAPH)*, 36(4), 2017.

[28] Yue Wang, Yongbin Sun, Ziwei Liu, Sanjay E. Sarma, Michael M. Bronstein, and Justin M. Solomon. Dynamic Graph CNN for Learning on Point Clouds. *arXiv:1801.07829 [cs]*, Jan. 2018.

[29] Yao Yao, Zixin Luo, Shiwei Li, Tian Fang, and Long Quan. MVSNet: Depth Inference for Unstructured Multi-view Stereo. *arXiv:1804.02505 [cs]*, Apr. 2018.

[30] Lequan Yu, Xianzhi Li, Chi-Wing Fu, Daniel Cohen-Or, and Pheng-Ann Heng. Pu-net: Point cloud upsampling network. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[31] Andrei Zaharescu, Edmond Boyer, and Radu Horaud. Transformesh: a topology-adaptive mesh-based approach to surface evolution. In *Asian Conference on Computer Vision*, pages 166–175. Springer, 2007.