# Fast-MVSNet: Sparse-to-Dense Multi-View Stereo With Learned Propagation and Gauss-Newton Refinement

Zehao Yu
ShanghaiTech University
yuzh@shanghaitech.edu.cn

Shenghua Gao
ShanghaiTech University
gaoshh@shanghaitech.edu.cn

## Abstract

*Almost all previous deep learning-based multi-view stereo (MVS) approaches focus on improving reconstruction quality. Besides quality, efficiency is also a desirable feature for MVS in real scenarios. Towards this end, this paper presents a Fast-MVSNet, a novel sparse-to-dense coarse-to-fine framework, for fast and accurate depth estimation in MVS. Specifically, in our Fast-MVSNet, we first construct a sparse cost volume for learning a sparse and high-resolution depth map. Then we leverage a small-scale convolutional neural network to encode the depth dependencies for pixels within a local region to densify the sparse high-resolution depth map. At last, a simple but efficient Gauss-Newton layer is proposed to further optimize the depth map. On one hand, the high-resolution depth map, the data-adaptive propagation method and the Gauss-Newton layer jointly guarantee the effectiveness of our method. On the other hand, all modules in our Fast-MVSNet are lightweight and thus guarantee the efficiency of our approach. Besides, our approach is also memory-friendly because of the sparse depth representation. Extensive experimental results show that our method is 5× and 14× faster than Point-MVSNet and R-MVSNet, respectively, while achieving comparable or even better results on the challenging Tanks and Temples dataset as well as the DTU dataset. Code is available at https://github.com/svip-lab/FastMVSNet.*

## 1. Introduction

Multi-view stereo (MVS) aims at recovering the dense 3D structure of a scene from a set of calibrated images. It is one of the fundamental problems in computer vision and has been extensively studied for decades, because of its wide applications in 3D reconstruction, augmented reality, autonomous driving, robotics, *etc* [1, 11].

The core of MVS is the dense correspondence across images. Traditional methods usually rely on hand-crafted photo-consistency metrics (*e.g.*, SSD, NCC). Designing a robust metric itself, however, is a challenging task, and thus some regularization techniques [25, 18] are required (*e.g.*, using MRF to enforce spatial consistency [25]). While these methods [34, 13] have shown impressive results, they are still incompetent on low-textured, specular, and reflective regions where local features are not discriminative for matching. Recent work [20, 43, 22, 23] shows that by using Deep CNNs, the performance of MVS can be further improved. For instance, in [43], an MVSNet is proposed, which builds a cost volume upon CNN features and uses 3D CNNs for cost volume regularization. Such an MVSNet significantly improves the overall 3D reconstruction quality compared to traditional hand-crafted metric based methods.

Nevertheless, all these deep learning-based methods use multi-scale 3D CNNs to predict the depth maps [43, 20, 21] or occupancy grids [22, 23], which are thus memory-consuming, as the memory requirement for 3D volume grows cubically. This restricts their application to high-resolution MVS. Therefore, some recent work [40, 33, 44, 7] has been proposed to address this memory-intensive issue. For instance, R-MVSNet [44] uses Convolutional GRU to replace 3D CNNs and thus reduces the memory requirement to quadratic, then a variational depth refinement is performed as a post-processing step to improve accuracy. Point-MVSNet [7] uses a coarse-to-fine strategy that first builds a relatively small 3D cost volume to predict a coarse depth map. Then a *PointFlow* module is used to upsample and refine the coarse results iteratively. Although these methods avoid the memory issue in MVS and achieve state-of-the-art 3D reconstruction quality on some challenging benchmark datasets, their efficiency is still far from satisfactory. In particular, R-MVSNet [44] needs 6.9 seconds to refine a depth map with size 400×300 and Point-MVSNet [7] uses around 3 seconds to refine a depth map of the size 640×480, which prohibits their application in large-scale scenarios. Besides 3D reconstruction quality, efficiency is also a desirable feature for MVS in real scenarios, which thus motivates us to work towards improving the efficiency of deep learning-based MVS methods.

Our observation is that a high-resolution depth map contains finer details, which would benefit the overall reconstruction. Directly predicting a high-resolution depth map from a 3D cost volume is, however, computationally expensive and memory-intensive. By contrast, a low-resolution depth can be predicted at much lower cost but with much fewer details. As a compromise, we propose to predict a sparse high-resolution depth map with low memory consumption first and then do the depth propagation to enrich the details with the reference image as a guidance. For depth propagation, motivated by the joint bilateral upsampling [26], we propose learning a small-scale convolution neural network to encode depth dependencies for pixels within a local region to densify the sparse high-resolution depth map. We further propose using a simple and fast Gauss-Newton layer, which takes deep CNN features of multi-view images and the coarse high-resolution depth map as inputs, to refine the dense high-resolution depth map. It is worth noting that all the modules we used are lightweight and small-scale and the resulting framework can be implemented efficiently. Meanwhile, all these modules are differentiable, and thus can be trained in an end-to-end manner. We therefore term our sparse-to-dense coarse-to-fine solution as Fast-MVSNet.

In summary, our contributions are as follows: i) We propose a novel sparse-to-dense coarse-to-fine framework for MVS, where the sparse-to-dense strategy guarantees the efficiency of our approach, and the coarse-to-fine strategy guarantees the effectiveness of our approach ; ii) We propose learning the depth dependencies for pixels within a local region with a small-scale convolutional neural network from the original image and use it to densify the sparse depth map. Meanwhile, such a network is motivated by joint bilateral upsampling. Thus the depth propagation procedure is explainable; iii) A differentiable Gauss-Newton layer is proposed to optimize the depth map, which enables our Fast-MVSNet to be end-to-end learnable; iv) Extensive experiments show that our method achieves better or comparable reconstruction results compared to other state-of-the-art methods while being much more efficient and memory-friendly. In particular, our method is 14$\times$ faster than R-MVSNet [44] and 5$\times$ faster than Point-MVSNet [7].

## 2. Related Work

### 2.1. Multi-view stereo reconstruction

Modern MVS algorithms usually use the following output scene representation: volume [22, 23, 27, 35], point cloud [28, 14, 12] or depth maps [37, 34, 13, 43, 44]. In particular, volumetric based methods discretize 3D space into regular grids and decide whether a voxel is near the surface. This is of high memory consumption, however, and is not scalable to large-scale scenarios. Point cloud based methods [28, 12] usually start from a sparse set of matched keypoints and use some propagation strategy to densify the point cloud. These methods, however, are difficult to be paralled as the propagation is proceeded sequentially. While depth map can be regarded as a special case of point cloud representation, (*e.g.*, pixel-wise point cloud), it reduces the reconstruction into the problem of per-view depth map estimation. Further, one can easily fuse the depth maps to point cloud [30] or volumetric reconstructions [31]. In this work, we also use the depth map representation. It is worth noting that our method shares some similarity with point cloud based methods where we start with a sparse depth map and learn to propagate the sparse depth map to a dense one with the help of reference image.

### 2.2. Learning-based MVS

Recently, with the power of representation learning of Deep CNNs, some researchers have proposed learning better patch representations, matchings and regularizations, demonstrating great success. In [16, 46, 17], researchers propose learning a similarity measurement between small image patches for matching cost computation. SurfaceNet [22] and DeepMVS [20] constuct a cost volume using multi-view images and use CNNs to learn the regularization of that cost volume. Yao *et al*. [43] propose an end-to-end MVS architecture that builds a cost volume upon CNN features and learns the cost volume regularization also with CNNs. However, the memory consumption for 3D cost volume grows cubically. Therefore, R-MVSNet [44] proposes using Convolutional GRU for cost volume regularization and thus avoid using memory-intensive 3D CNNs. By contrast, Point-MVSNet [7] uses a coarse-to-fine stategy that first predicts a low resolution depth map and iteratively upsamples and refines the depth map. While these method have shown impressive results, their efficiency is still far from satisfactory. Our work is mostly related to Point-MVSNet [7] as we also use a coarse-to-fine strategy. Instead of using a time-consuming strategy to sample depth hypotheses for refinement, however, we learn to directly optimize the depth map with a differentiable Gauss-Newton layer, which is efficient and ensures our network can be trained in an end-to-end manner.

### 2.3. Depth map upsampling and propagation

Upsampling and propagation are ubiquitous tools in computer vision as we typically compute a low resolution result with low computational cost and interploate the result to obtain a high resolution result. Simple upsampling methods such as nearest neighbour and bilinear interpolation, however, subsequently suffer from over smoothing around image edges. Instead, by using high resolution image as a guidance, joint bilateral upsampling [26, 3] can preserve edge characteristics. Xu *et al*. [42] further propose using

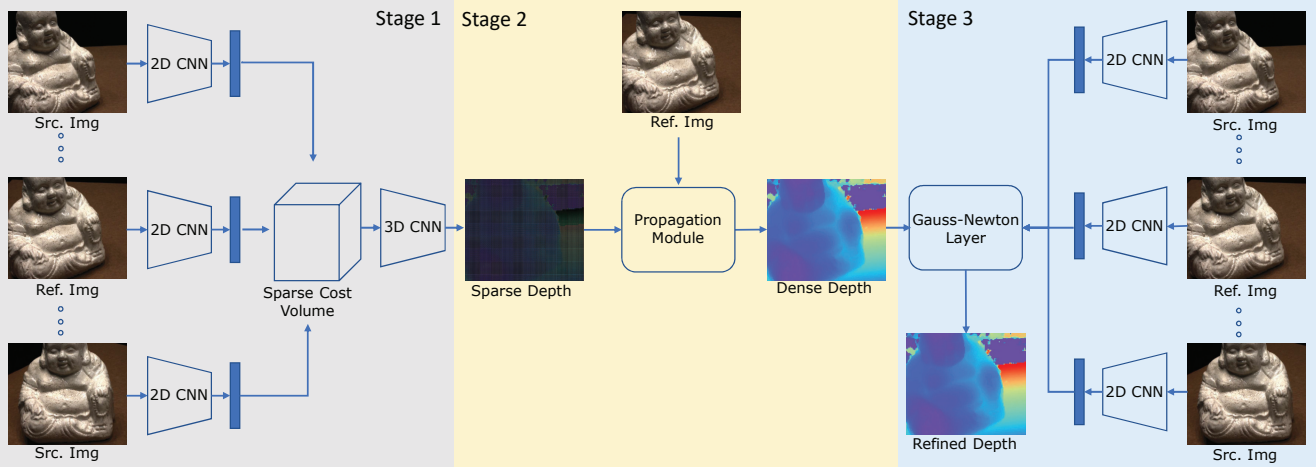



Figure 1: Network architecure of the Fast-MVSNet. In the **first stage**, we construct a sparse cost volume upon 2D CNN features and predict a sparse high-resolution depth map using a 3D CNN. In the **second stage**, we design a simple but efficient network to propagate the sparse depth map to a dense depth map. In the **third stage**, we propose a differentiable Gauss-Newton layer to further refine the depth map.

multi-view geometric consistency as a guidance for depth map upsampling, while Wei *et al*. [41] extend joint bilateral upsamling to incorporate surface normal information. These methods, however, rely on hand-crafted strategy and their kernel parameters need to be manually tuned. Unlike these methods, we propose learning the propagation of our sparse depth map with image guidance and show that by incorporating a learnable propagation module, reconstruction results can be further improved.

### 2.4. Learning-based optimization

Some recent work has been proposed to learn the optimization of nonlinear least square objective functions by utilizing the differentiable nature of iterative optimization algorithms. These optimization algorithms are unrolled for a fixed number of iterations, and each iteration is implemented as a layer in a neural network. In [8], an LSTM [19] is used to model the Levenberg-Marquardt (LM) algorithm and predicts the update at each step directly. In [36], Tang *et al*. propose a differentiable LM algorithm by learning to predict the damping factor of standard LM algorithm, while Lv *et al*. [29] use learnable modules to replace multiple components of the inverse compositional algorithm [2]. Unlike these methods, CodeSLAM [4] and SceneCode [47] learn a compact representation (*i.e*., *code*) of scene for later optimization, while Stumberg *et al*. [39] propose a Gauss-Newton loss to learn robust representation of images under different weather conditions. Our method is particularly inspired by this line of work where we propose a differentiable Gauss-Newton layer for efficient depth map refinement but our method is not restricted to scene dependent depth basis [36] or learned code [4, 47].

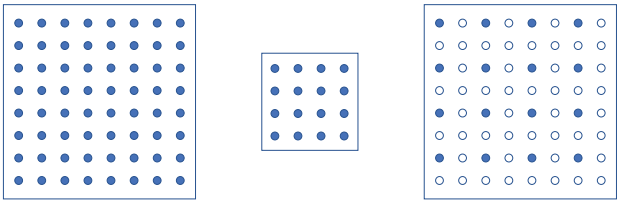
## 3. Method

Our goal is to design an effective and efficient framework for MVS. Following recent successes [20, 43, 44, 7, 21], we use per-view depth map as scene representation for its flexibility and scalability. That is to estimate a depth map for a reference image $I_0$ given a set of neighboring source images $\{I_i\}_{i=1}^{N}$.

To this end, we propose a Fast-MVSNet, an efficient MVS framework that ultizes a sparse-to-dense coarse-to-fine strategy for depth map estimation. Specifically, we **first** estimate a sparse high-resolution depth map such that existing MVS methods can be applied at a lower cost (*i.e*., less computational cost and less memory consumption). **Then** we design a simple but efficient propagation module to propagate the sparse depth map to a dense depth map. **Finally**, a differentiable Gauss-Newton layer is proposed to further optimize the depth map for sub-pixel accuracy. The overall pipeline of our method is shown in Figure 1. Next, we will introduce each component of our method in details.

### 3.1. Sparse high-resolution depth map prediction

Our first step is to estimate a sparse high-resolution depth map for the reference image $I_0$. Figure 2 shows the key differences between our sparse depth map representation and depth maps in other methods. We estimate a sparse high-resolution depth map with low memory and computation costs while other methods either estimate a high-resolution depth map [20, 43] with high memory costs or a low-resolution depth map [7] without fine details. We argue that our sparse high-resolution representation is more adequate than a low-resolution representation because: i)



(a) (b) (c)

Figure 2: Initialization of depth map. (a) MVSNet [43] and R-MVSNet [44]. (b) PointMVSNet [7]. (c) Ours. Unlike other methods, we estimate a sparse high-resolution depth map considering efficiency and quality.

training with a low-resolution depth map requires downsampling the ground-truth depth map accordingly. If we downsample the ground-truth depth map with the nearest neighbour method, then the low-resolution representation is the same as our sparse high-resolution representation. In this case, however, the resulting depth map is not well aligned with the extracted low-resolution feature map. If we use bilinear interpolation for downsampling, it will cause artifacts around regions with depth discontinuities; ii) fine details are lost in the low-resolution depth map. Recovering a high-resolution depth map with fine details from a low-resolution one requires non-trival and complicated upscaling methods [10].

To predict our sparse high-resolution depth map, we adapt the MVSNet [43] for our task. Specifically, we first use the same 8-layer 2D CNN as MVSNet to extract image features, then we build a sparse cost volume in the frustum of the reference image. Finally, we use 3D CNN to regularize the cost volume and predict a sparse depth map via differentiable argmax [43]. Our method is a general framework. While we use 3D CNN for cost volume regularization, other regularization methods such as Convolutional GRU [44] are also applicable.

We highlight the differences of our cost volume with previous methods [43, 7] as follows: i) our cost volume is of size $\frac{1}{8}H \times \frac{1}{8}W \times N \times F$, while MVSNet use a cost volume of size $\frac{1}{4}H \times \frac{1}{4}W \times N \times F$, where $N$ is the number of depth planes and $F$ is the number of feature channels; ii) MVSNet uses 256 virtual depth planes, while we use the same number of depth planes as that in Point-MVSNet [7]. In particular, we use 48 and 96 virtual depth planes for training and evaluation respectively; iii) We use an 8-layer 2D CNN to extract image features with $F = 32$ channels while Point-MVSNet [7] uses an 11-layer 2D CNN to extract image features with $F = 64$ channels. As a result, the memory usage of our cost volume is $\frac{1}{2}$ of that in Point-MVSNet [7].

Interestingly, due to our sparse representation, the 3D CNN acts like dilated convolutions [6] with dilation 2 in the spatial domain. Thus it has the potential to incorporate larger spatial contextual information for regularization.

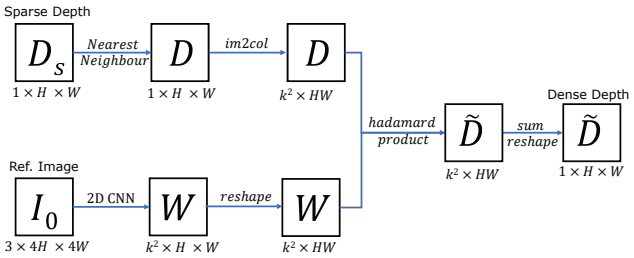



Figure 3: A diagram of the propagation module.

### 3.2. Depth map propagation

The former step provides us a high-resolution but sparse depth map $D$. We now need to propagate the sparse depth map to obtain a dense depth map $\tilde{D}$. A simple strategy is to use nearest neighbour for this purpose. This nearest neighbor method, however, does not consider the original image information and thus may not work well around depth boundaries. Another natural choice is the joint bilateral upsampler [26, 3, 41] which uses the information of original high-resolution image as a guidance. Formally, it takes the following form:

$$\tilde{D}(p) = \frac{1}{z_p} \sum_{q \in N(p)} D(q) f(\|p - q\|) g(\|I_p - I_q\|) \quad (1)$$

where $f$ is the spatial filter kernel, $g$ is the range filter kernel, $N(p)$ is the local $k \times k$ neighbour around position $p$ and $z_p$ is a normalization term. These two kernel parameters, however, may be different for diverse scenes and need to be manually tuned.

We therefore propose replacing $f(\|p - q\|)g(\|I_p - I_q\|)$ with a weight $w_{p,q}$ and learning the weights with a simple network. Mathematically, we use the following form:

$$\tilde{D}(p) = \frac{1}{z_p} \sum_{q \in N(p)} D(q) \cdot w_{p,q} \quad (2)$$

where $w_{p,q}$ is the output of a CNN and is learned in a data-driven manner. We note that while we do not explicitly account for spatial information, it is indeed implicitly encoded by the network. Further, as we predict different weights for different position $p$, our method can be viewed as a generalization of the standard bilateral upsampler that applies a fixed kernel for every position $p$.

**Implementation.** The sparse depth map $D_s$ is first propagate to a dense depth map $D$ using nearest neighbour. In parallel, a CNN takes the reference image $I_0$ as input and outputs $k \times k$ weights $W$ for each position. Finally, the propagated depth map $\tilde{D}$ is computed using Equation 2. Note that the computation of Equation 2 can be efficiently implemented using vectorization (*i.e. im2col*). The details of the proposed propagation module are shown in Figure 3.

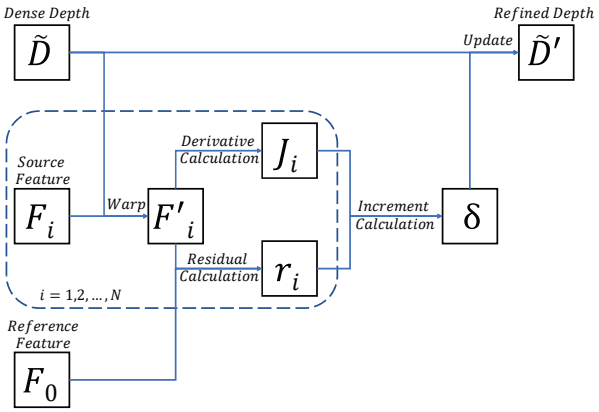



Figure 4: A diagram of the differentiable Gauss-Newton Layer. We ignore camera parameters here for simplicity.

To predict the weight $W$, we simply use the same network architecture as that in MVSNet to extract image features and append a two-layer 3×3 convolutional network to predict a feature map with $k \times k$ channels. The softmax function is applied in the channel dimension for normalization.

### 3.3. Gauss-Newton refinement

As we concentrate on the efficient inference of a dense high-resolution depth map in the previous step, the accuracy of the resulting depth map is insufficient. Therefore, we propose using the Gauss-Newton algorithm to refine the depth map. While there are various methods could be used for depth map refinement [44, 43, 7], we select the Gauss-Newton algorithm for its efficiency.

Mathematically, given a point $p$ with depth $\tilde{D}_p$ in the reference image, we aim to minimize the following error function:

$$E(p) = \sum_{i=1}^{N} \|F_i(p'_i) - F_0(p)\|_2 \tag{3}$$

where $F_i$ and $F_0$ are the deep representation of the source image $I_i$ and the reference image $I_0$, respectively, $p'_i$ is the reprojected point of $p$ in image $I_i$ and $F_i(p)$ corresponds to the features at $p$ in $F_i$. $p'_i$ can be computed as

$$p'_i = K_i(R_i R_0^{-1}(\tilde{D}(p)K_0^{-1}p - t_0) + t_i) \tag{4}$$

where $\{K_i, R_i, t_i\}_{i=0}^{N}$ denote the camera intrinsics, rotations and translations of corresponding images.

We apply the Gauss-Newton algorithm to minimize $E(p)$. Specifically, starting with an initial depth $\tilde{D}_p$, we compute the residual $r_i(p)$ of $p$ for each source image $I_i$:

$$r_i(p) = F_i(p'_i) - F_0(p) \tag{5}$$

Then for each residual $r_i(p)$, we compute their first order derivative with respect to $\tilde{D}(p)$ as:

$$J_i(p) = \frac{\partial F_i(p'_i)}{\partial p'_i} \cdot \frac{\partial p'_i}{\partial \tilde{D}(p)} \tag{6}$$

Finally, we can obtain the increment $\delta$ to the current depth as:

$$\delta = -(J^T J)^{-1} J^T r \tag{7}$$

where $J$ is the stack of jacobians $\{J_i(p)\}_{i=1}^{N}$, and $r$ is the stack of residual vectors $\{r_i(p)\}_{i=1}^{N}$. Therefore, the refined depth is:

$$\tilde{D}'(p) = \tilde{D}(p) + \delta. \tag{8}$$

Further, the Gauss-Newton algorithm is naturally differentiable and can be implemented as a layer in a neural network without additional learnable parameters. As shown in Figure 4, the Gauss-Newton layer takes multi-view image features, camera parameters and an initial depth map as input, then outputs a refined depth map. The overall network can be trained in an end-to-end manner. Therefore, it can learn suitable features for efficient optimization [36]. We find that it converges quickly with only a single step of update. Note that our Gauss-Newton layer is different with the refinement of R-MVSNet which use gradient decent to optimize hard-crafted photo-consistency metric, whereas we intergrate the optimization in training. Furthermore, as we do not need to sample depth hypotheses, our method is more efficient and memory-friendly compared to Point-MVSNet [7].

### 3.4. Training Loss

Following previous methods [43, 7], we use the mean absolute difference between the estimated depth map and ground truth depth map as our training loss. Both the initial depth map $\tilde{D}$ and the refined depth map $\tilde{D}'$ are included in our trianing loss:

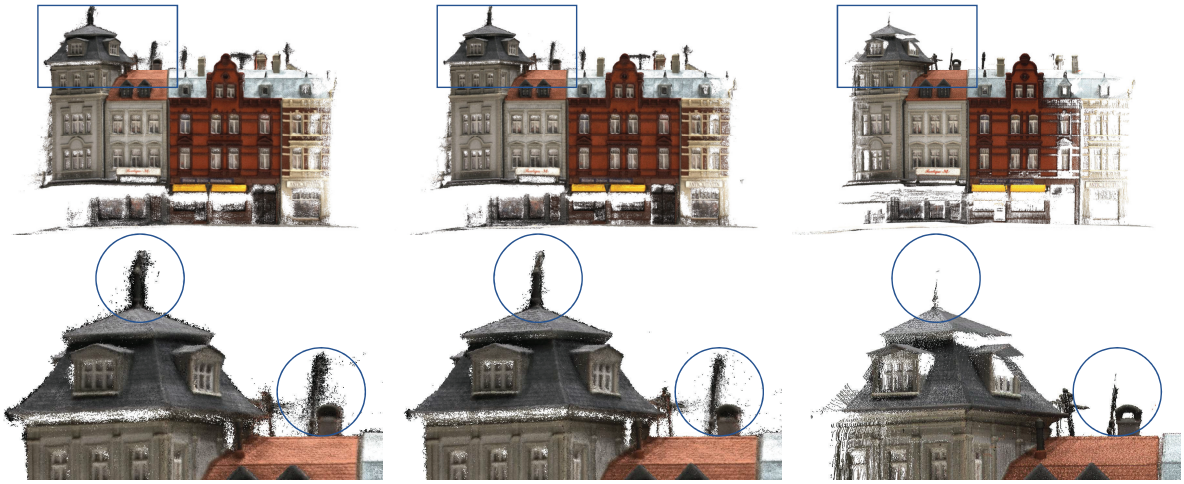
$$Loss = \sum_{p \in \mathbf{p}_{valid}} \|\tilde{D}(p) - \hat{D}(p)\| + \lambda \cdot \|\tilde{D}'(p) - \hat{D}(p)\| \tag{9}$$

where $\hat{D}$ is the ground truth depth map, $\mathbf{p}_{valid}$ denotes the set of valid ground truth depths and $\lambda$ is the weight that balances the two losses. We set $\lambda$ to be 1.0 in all the experiments.

## 4. Experiments

### 4.1. The DTU dataset

The DTU dataset [1] is a large scale MVS dataset, which contains 80 scenes with large diversity. Each scene is captured at 49 or 64 precise camera positions with 7 different lighting conditions. The dataset provides reference models which are acquired by an accurate structured light scanner along with high-resolution RGB images. We use the same training, validation and evaluation sets as that in other learning based methods [22, 43, 44, 7].



Point-MVSNet [7] Ours Ground Truth

Figure 5: Qualitative results of *scan9* of the DTU dataset. Top: Whole point cloud. Bottom: Zoomed local region. We use the same point cloud fusion parameters with Point-MVSNet. As shown in blue circle region, our reconstruction contains less noise around fine detailed structures, which demonstrates the effectiveness of our method.

| | Acc. (mm) | Comp. (mm) | Overall (mm) |
|---|---|---|---|
| Camp [5] | 0.835 | 0.554 | 0.695 |
| Furu [12] | 0.613 | 0.941 | 0.777 |
| Tola [38] | 0.342 | 1.190 | 0.766 |
| Gipuma [13] | **0.283** | 0.873 | 0.578 |
| PU-Net [45] | 1.220 | 0.667 | 0.943 |
| SurfaceNet [22] | 0.450 | 1.040 | 0.745 |
| MVSNet [43] | 0.396 | 0.527 | 0.462 |
| R-MVSNet [44] | 0.385 | 0.452 | 0.417 |
| PointMVSNet [7] | 0.361 | 0.421 | 0.391 |
| Ours | 0.336 | **0.403** | **0.370** |

Table 1: Quantitative results of reconstruction quality on the DTU evaluation data [1]. Our method outperforms all methods in terms of reconstruction completeness and overall quality.

### 4.2. Implementation Details

**Training.** We use the training data generated by MVSNet [43]. The point cloud provided by the DTU dataset are used to reconstruct mesh surfaces which are then used to render depth maps for training. We implement our model with PyTorch [32]. We set the resolution of input image to 640×512 and the number of views $N$ to 3. To choose source images for training, the same view selection strategy as MVSNet [43] is used. We set the number of depth planes $D = 48$ in sparse depth map prediction where depth hypotheses are uniformly sampled from $425mm$ to $921mm$. Following PointMVSNet [7], we use the RMSProp optimizer with initial learning rate 0.0005 and decrease the learning rate by 0.9 every 2 epochs. The batch size is set to 16 on 4 NVIDIA GTX 2080Ti GPU devices. We first pretrain the sparse depth map prediction module and propagation module for 4 epochs. Then the overall model is trained end-to-end for another 12 epochs. Details of network architecture are described in supplementary material.

**Testing.** After the propagation of sparse depth map, we get a dense depth map of size $1 \times H \times W$. For a fair comparison with Point-MVSNet [7], we upsample the depth map to $2 \times H \times W$ with nearest neighbor before Gauss-Newton refinement. We use $N = 5$ images with resolution 1280×960 as input and we set the number of depth planes $D = 96$. We first predict a depth map for each reference image and then use the post processing provided by [43] to fuse the predicted depth maps into point cloud. The same parameters for depth map fusion as Point-MVSNet [7] are used unless otherwise specified.

### 4.3. Results on the DTU dataset

We compare our method with both traditional methods and recent learning based methods. The quantitative results are shown in Table 1. While Gipuma [13] achieves the best performance in terms of *Accuracy*, our method outperforms all competing methods in both *Completeness* and *Overall* quality. Figure 5 shows the qualitative comparison with the results of Point-MVSNet. Our reconstruction is cleaner around fine detailed structures, which validates the effectiveness of our method.

We further demonstrate the efficiency and the effectiveness of the proposed method by comparing the reconstruction quality, depth map resolution, GPU memory requirements and runtime with state-of-the-art methods in Table 2. For a fair comparison with Point-MVSNet [7], the runtime is measured on an NVIDIA GTX 1080Ti GPU. As shown in Table 2, our method outperforms all methods in terms of all evaluation metrics while being more efficient and more memory-friendly. In particular, our method is about 2× faster than MVSNet [43], 14× faster than R-MVSNet [44]

(a): Sparse High-Resolution (b): (a) + Propagation Module (c): (b) + Gauss-Newton Layer Ground Truth

Figure 6: Qualitative results of *scan12* of the DTU dataset. Top: whole point cloud. Middle and bottom: zoomed local region of rectangle. Reconstruction results become denser and more detailed when gradually adding propagation module and Gauss-Newton layer (see text region).

| | Acc. (mm) | Comp. (mm) | Overall (mm) | Depth Map Res. | GPU Mem. (GB) | Runtime (s) |
|---|---|---|---|---|---|---|
| MVSNet[43] | 0.456 | 0.646 | 0.551 | 288×216 | 10.8 | 1.05 |
| R-MVSNet[43] | 0.385 | 0.452 | 0.417 | 400×300 | 6.7 | 9.1 |
| Point-MVSNet [7] | 0.361 | 0.421 | 0.391 | **640×480** | 8.7 | 3.35 |
| Ours | **0.336** | **0.403** | **0.370** | **640×480** | **5.3** | **0.6** |

Table 2: Comparison results measured by reconstruction quality, depth map resolution, GPU memory requirements and runtime on the DTU evaluation set. The result of MVSNet [43] is quoted from Point-MVSNet [7]. Due to the GPU memory limitation, the resolution of MVSNet [43] is decreased to 1152×864×192. Our method outperforms all methods in terms of all evaluation metrics while being more efficient and more memory-friendly.

and 5× faster than Point-MVSNet [7].

### 4.4. Ablation study

Due to space limitations, we refer readers to supplementary material for additional ablation studies, including Gauss-Newton refinements with more iterations and depth map fusion with different parameters.

**Effectiveness of the sparse high-resolution depth map.** To evaluate the effectiveness of our sparse high-resolution depth map representation, we compare the reconstruction results with the low-resolution depth map representation in Table 4. For a fair comparison, both the low-resolution depth map and our sparse high-resolution depth map are upsampled to the 640× 480 with nearest neighbor. As shown in the first two rows in Table 4, our sparse high-resolution depth map achieve better results.

**Effectiveness of propagation module.** To evaluate the effectiveness of our learned propagation module, we show the results with or without the propagation module in the second and third row in Table 4, showing the propagation module can further improved the reconstruction results.

**Effectiveness of the Gauss-Newton Refinement.** We compare the results with or without the Gauss-Newton refinement in the third and fouth row in Table 4. With the Gauss-Newton refinement, the relative improvement of *Overall* reconstruction quality is 9.5% (from 0.409 to 0.370), showing the effectiveness of our Gauss-Newton Refinement.

**Efficiency of the Gauss-Newton Refinement.** Both our Gauss-Newton layer and the *PointFlow* module proposed in Point-MVSNet [7] aim to refine a coarse depth map. *PointFlow* uses a hypothesis testing strategy that first samples a set of hypotheses (around the current depth prediction) and uses a network to select a better hypothesis via weighted average among all hypotheses. In the contrary, we formulate the depth map refinement as an optimization problem and intergate the optimization into an end-to-end framework. Compared with hypotheses sampling solution in Point-MVSNet, our formulation is simple and more efficient.

In order to evaluate the efficiency of the proposed Gauss-Newton layer, we replace the *PointFlow* Module of Point-MVSNet with our differentiable Gauss-Newton layer and train the network from scratch. The comparison of reconstruction results are shown in the last two rows in Table 4. We achieve comparable results with Point-MVSNet while

Figure 7: Reconstruction results of *intermediate set* in the *Tank and Temples* dataset [24]. Our method can reconstruct dense and visually appealing complex scenes.

| | Mean | Family | Francis | Horse | Lighthouse | M60 | Panther | Playground | Train |
|---|---|---|---|---|---|---|---|---|---|
| MVSNet [43] | 43.48 | 55.99 | 28.55 | 25.07 | **50.79** | **53.96** | **50.86** | 47.90 | 34.69 |
| R-MVSNet [44] | **48.40** | **69.96** | **46.65** | 32.59 | 42.95 | 51.88 | 48.80 | 52.00 | 42.38 |
| Point-MVSNet [7] | 48.27 | 61.79 | 41.15 | 34.20 | **50.79** | 51.97 | 50.85 | 52.38 | **43.06** |
| Ours | 47.39 | 65.18 | 39.59 | **34.98** | 47.81 | 49.16 | 46.20 | **53.27** | 42.91 |

Table 3: Evaluation results on *Tanks and Temples* benchmark [24]. We achieve comparable results with state-of-the-art methods.

| method | Acc. | Comp. | Overall |
|---|---|---|---|
| low res. | 0.517 | 0.557 | 0.537 |
| sparse high res. | 0.394 | 0.478 | 0.436 |
| sparse high res. + prop. | 0.370 | 0.448 | 0.409 |
| sparse high res. + prop. + GN | **0.336** | **0.403** | **0.370** |
| low res. + *PointFlow* [7] | 0.361 | 0.421 | 0.391 |
| low res. + GN. | 0.376 | 0.417 | 0.396 |

Table 4: Ablation study on the DTU evaluation dataset, which demonstrates the effectiveness of different components of our method. Low res. denotes low-resolution depth map, sparse high res. denotes our sparse high-resolution depth map, prop. denotes propagation module and GN denotes Gauss-Newton refinement.

our method is 5× faster. Further, as we directly optimize the depth instead of sampling possible depth hypotheses, our method is more memory-friendly and does not need to adopt a divide and conquer strategy to refine a high-resolution depth map (*e.g.* 640×480).

We show the comparison of recontruction results when adding different components of our method in Figure 6. The results become denser and contains much finer details especilly in the text region.

### 4.5. Generalization

To evaluate the generalizability of our proposed method, we test it on the large scale *Tanks and Temples* dataset [24]. We use the model trained on the DTU dataset without fine-tuning for testing. We use $N = 5$ images with resolution 1920×1056 as input. We set the number of depth planes $D = 96$. We use the camera parameters provided by MVSNet [43] for a fair comparison. The evaluation results are shown in Table 2. We achieve comparable results with state-of-the-art methods, which demonstrates the generalizability of the proposed method. Qualitative results are shown in Figure 7. Our reconstruction is dense and visually apearling.

## 5. Conclusion

We propose Fast-MVSNet as an efficient MVS solution, which leverages a sparse-to-dense coarse-to-fine strategy. We first estimate a sparse high-resolution depth map at a lower costs. Then the sparse high-resolution depth map is propagated to a dense depth map via a simple propagation module. Finally, a differentiable Gauss-Newton layer is proposed to optimize the depth map further. Experimental results on two challenging datasets verify the effectiveness and efficiency of the proposed method.

## Acknowledgements

The work was supported by National Key R&D Program of China (2018AAA0100704), NSFC #61932020, and ShanghaiTech-Megavii Joint Lab. We would like to thank Dongze Lian, Weixin Luo, Lei Jin and Shenhan Qian for their insightful comments during the preparation of the manuscript.

## References


[1] Henrik Aanæs, Rasmus Ramsbøl Jensen, George Vogiatzis, Engin Tola, and Anders Bjorholm Dahl. Large-scale data for multiple-view stereopsis. *International Journal of Computer Vision*, pages 1–16, 2016. 1, 5, 6, 9, 10, 11

[2] Simon Baker and Iain Matthews. Lucas-kanade 20 years on: A unifying framework. *International journal of computer vision*, 56(3):221–255, 2004. 3

[3] Jonathan T Barron and Ben Poole. The fast bilateral solver. In *European Conference on Computer Vision*, pages 617–632. Springer, 2016. 2, 4

[4] Michael Bloesch, Jan Czarnowski, Ronald Clark, Stefan Leutenegger, and Andrew J Davison. Codeslam—learning a compact, optimisable representation for dense visual slam. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2560–2568, 2018. 3

[5] 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. 6

[6] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *IEEE transactions on pattern analysis and machine intelligence*, 40(4):834–848, 2017. 4

[7] Rui Chen, Songfang Han, Jing Xu, and Hao Su. Point-based multi-view stereo network. In *The IEEE International Conference on Computer Vision (ICCV)*, 2019. 1, 2, 3, 4, 5, 6, 7, 8, 9

[8] Ronald Clark, Michael Bloesch, Jan Czarnowski, Stefan Leutenegger, and Andrew J. Davison. Learning to solve nonlinear least squares for monocular stereo. In *The European Conference on Computer Vision (ECCV)*, September 2018. 3

[9] Simon Donne and Andreas Geiger. Learning non-volumetric depth fusion using successive reprojections. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019. 9

[10] Alexey Dosovitskiy, Philipp Fischer, Eddy Ilg, Philip Hausser, Caner Hazirbas, Vladimir Golkov, Patrick Van Der Smagt, Daniel Cremers, and Thomas Brox. Flownet: Learning optical flow with convolutional networks. In *Proceedings of the IEEE international conference on computer vision*, pages 2758–2766, 2015. 4

[11] Yasutaka Furukawa, Carlos Hernández, et al. Multi-view stereo: A tutorial. *Foundations and Trends® in Computer Graphics and Vision*, 9(1-2):1–148, 2015. 1

[12] 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. 2, 6

[13] Silvano Galliani, Katrin Lasinger, and Konrad Schindler. Massively parallel multiview stereopsis by surface normal diffusion. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 873–881, 2015. 1, 2, 6

[14] Michael Goesele, Noah Snavely, Brian Curless, Hugues Hoppe, and Steven M Seitz. Multi-view stereo for community photo collections. In *2007 IEEE 11th International Conference on Computer Vision*, pages 1–8. IEEE, 2007. 2

[15] Lei Han, Mengqi Ji, Lu Fang, and Matthias Nießner. Regnet: Learning the optimization of direct image-to-image pose registration. *arXiv preprint arXiv:1812.10212*, 2018.

[16] 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. 2

[17] Wilfried Hartmann, Silvano Galliani, Michal Havlena, Luc Van Gool, and Konrad Schindler. Learned multi-patch similarity. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1586–1594, 2017. 2

[18] Heiko Hirschmuller. Stereo processing by semiglobal matching and mutual information. *IEEE Transactions on pattern analysis and machine intelligence*, 30(2):328–341, 2007. 1

[19] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997. 3

[20] Po-Han Huang, Kevin Matzen, Johannes Kopf, Narendra Ahuja, and Jia-Bin Huang. Deepmvs: Learning multi-view stereopsis. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2821–2830, 2018. 1, 2, 3

[21] Sunghoon Im, Hae-Gon Jeon, Stephen Lin, and In So Kweon. Dpsnet: end-to-end deep plane sweep stereo. *arXiv preprint arXiv:1905.00538*, 2019. 1, 3

[22] Mengqi Ji, Juergen Gall, Haitian Zheng, Yebin Liu, and Lu Fang. Surfacenet: An end-to-end 3d neural network for multiview stereopsis. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2307–2315, 2017. 1, 2, 5, 6

[23] Abhishek Kar, Christian Häne, and Jitendra Malik. Learning a multi-view stereo machine. In *Advances in neural information processing systems*, pages 365–376, 2017. 1, 2

[24] Arno Knapitsch, Jaesik Park, Qian-Yi Zhou, and Vladlen Koltun. Tanks and temples: Benchmarking large-scale scene reconstruction. *ACM Transactions on Graphics*, 36(4), 2017. 8

[25] Vladimir Kolmogorov and Ramin Zabih. Computing visual correspondence with occlusions via graph cuts. Technical report, Cornell University, 2001. 1

[26] Johannes Kopf, Michael F Cohen, Dani Lischinski, and Matt Uyttendaele. Joint bilateral upsampling. In *ACM Transactions on Graphics (ToG)*, volume 26, page 96. ACM, 2007. 2, 4

[27] Kiriakos N Kutulakos and Steven M Seitz. A theory of shape by space carving. *International journal of computer vision*, 38(3):199–218, 2000. 2

[28] 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. 2

[29] Zhaoyang Lv, Frank Dellaert, James M Rehg, and Andreas Geiger. Taking a deeper look at the inverse compositional algorithm. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4581–4590, 2019. 3

[30] Paul Merrell, Amir Akbarzadeh, Liang Wang, Philippos Mordohai, Jan-Michael Frahm, Ruigang Yang, David Nistér, and Marc Pollefeys. Real-time visibility-based fusion of depth maps. In *2007 IEEE 11th International Conference on Computer Vision*, pages 1–8. IEEE, 2007. 2

[31] Richard A Newcombe, Shahram Izadi, and Otmar Hilliges. Kinectfusion: Real-time dense surface mapping and tracking. 2011. 2

[32] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in PyTorch. In *NeurIPS Workshop*, 2017. 6

[33] 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. 1

[34] Johannes L Schönberger, Enliang Zheng, Jan-Michael Frahm, and Marc Pollefeys. Pixelwise view selection for unstructured multi-view stereo. In *European Conference on Computer Vision*, pages 501–518. Springer, 2016. 1, 2

[35] Steven M Seitz and Charles R Dyer. Photorealistic scene reconstruction by voxel coloring. *International Journal of Computer Vision*, 35(2):151–173, 1999. 2

[36] Chengzhou Tang and Ping Tan. Ba-net: Dense bundle adjustment network. *arXiv preprint arXiv:1806.04807*, 2018. 3, 5

[37] Engin Tola, Christoph Strecha, and Pascal Fua. Efficient large-scale multi-view stereo for ultra high-resolution image sets. *Machine Vision and Applications*, 23(5):903–920, 2012. 2

[38] 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. 6

[39] Lukas von Stumberg, Patrick Wenzel, Qadeer Khan, and Daniel Cremers. Gn-net: The gauss-newton loss for deep direct slam. *arXiv preprint arXiv:1904.11932*, 2019. 3

[40] 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 (TOG)*, 36(4):72, 2017. 1

[41] Mengqiang Wei, Qingan Yan, Fei Luo, Chengfang Song, and Chunxia Xiao. Joint bilateral propagation upsampling for unstructured multi-view stereo. *The Visual Computer*, 35(6-8):797–809, 2019. 3, 4

[42] Qingshan Xu and Wenbing Tao. Multi-scale geometric consistency guided multi-view stereo. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5483–5492, 2019. 2

[43] Yao Yao, Zixin Luo, Shiwei Li, Tian Fang, and Long Quan. Mvsnet: Depth inference for unstructured multi-view stereo. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 767–783, 2018. 1, 2, 3, 4, 5, 6, 7, 8, 9

[44] Yao Yao, Zixin Luo, Shiwei Li, Tianwei Shen, Tian Fang, and Long Quan. Recurrent mvsnet for high-resolution multi-view stereo depth inference. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5525–5534, 2019. 1, 2, 3, 4, 5, 6, 8

[45] 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. 6

[46] Jure Zbontar, Yann LeCun, et al. Stereo matching by training a convolutional neural network to compare image patches. *Journal of Machine Learning Research*, 17(1-32):2, 2016. 2

[47] Shuaifeng Zhi, Michael Bloesch, Stefan Leutenegger, and Andrew J. Davison. Scenecode: Monocular dense semantic reconstruction using learned encoded scene representations. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019. 3