PatchmatchNet: Learned Multi-View Patchmatch Stereo

Fangjinhua Wang[1] Silvano Galliani[2] Christoph Vogel[2] Pablo Speciale[2] Marc Pollefeys[1,2]
[1]Department of Computer Science, ETH Zurich
[2]Microsoft Mixed Reality & AI Zurich Lab

Abstract

We present PatchmatchNet, a novel and learnable cascade formulation of Patchmatch for high-resolution multi-view stereo. With high computation speed and low memory requirement, PatchmatchNet can process higher resolution imagery and is more suited to run on resource limited devices than competitors that employ 3D cost volume regularization. For the first time we introduce an iterative multi-scale Patchmatch in an end-to-end trainable architecture and improve the Patchmatch core algorithm with a novel and learned adaptive propagation and evaluation scheme for each iteration. Extensive experiments show a very competitive performance and generalization for our method on DTU, Tanks & Temples and ETH3D, but at a significantly higher efficiency than all existing top-performing models: at least two and a half times faster than state-of-the-art methods with twice less memory usage. Code is available at `https://github.com/FangjinhuaWang/PatchmatchNet`.

1. Introduction

Given a collection of images with known camera parameters, multi-view stereo (MVS) describes the task of reconstructing the dense geometry of the observed scene. Despite being a fundamental problem of geometric computer vision that has been studied for several decades, MVS is still a challenge. This is due to a variety of de-facto unsolved problems occurring in practice such as occlusion, illumination changes, untextured areas and non-Lambertian surfaces [1, 24, 32].

The success of Convolutional Neural Networks (CNN) in almost any field of computer vision ignites the hope that data driven models can solve some of these issues that classical MVS models struggle with. Indeed, many learning-based methods [6, 28, 39, 42, 43] appear to fulfill such promise and outperform some traditional methods [16, 31] on MVS benchmarks [1, 24]. While being successful at the benchmark level, most of them do only pay limited attention to scalability, memory and run-time. Currently, most

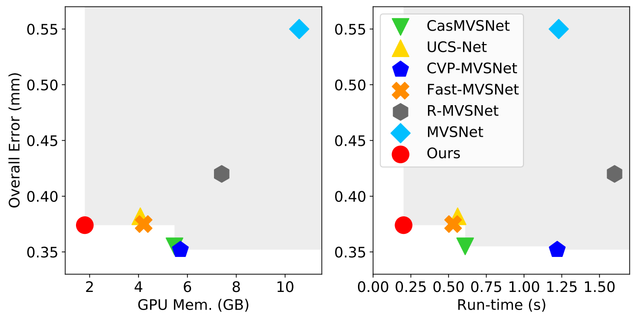

Figure 1: Comparison with state-of-the-art learning-based multi-view stereo methods [7,17,41,42,43,44] on DTU [1]. Relationship between error, GPU memory and run-time with image size 1152×864.

learning-based MVS methods [6, 28, 39, 42] construct a 3D cost volume, regularize it with a 3D CNN and regress the depth. As 3D CNNs are usually time and memory consuming, some methods [39, 42] down-sample the input during feature extraction and compute both, the cost volume and the depth map at low-resolution. Yet, according to Fig. 1, delivering depth maps at low resolution can harm accuracy. Methods that do not scale up well to realistic image sizes of several mega-pixel cannot exploit the full resolution due to memory limitations. Evidently, low memory and time consumption are key to enable processing on memory and computational restricted devices such as phones or mixed reality headsets, as well as in time critical applications. Recently, researchers tried to alleviate these limitations. For example, R-MVSNet [43] decouples the memory requirements from the depth range and sequentially processes the cost volume at the cost of an additional runtime penalty. [7, 17, 41] include cascade 3D cost volumes to predict high-resolution depth map from coarse to fine with high efficiency in time and memory.

Several traditional MVS methods [16, 31, 38, 45] abandon the idea of holding a structured cost volume completely and instead are based on the seminal Patchmatch [2] algorithm. Patchmatch adopts a randomized, iterative algorithm for approximate nearest neighbor field computation [2]. In particular, the inherent spatial coherence of depth maps is exploited to quickly find a good solution without the need to look through all possibilities. Low memory requirements –

independent of the disparity range – and an implicit smoothing effect make this method very attractive for our deep learning based MVS setup.

In this work, we propose PatchmatchNet, a novel cascade formulation of learning-based Patchmatch, which aims at decreasing memory consumption and run-time for high-resolution multi-view stereo. It inherits the advantages in efficiency from classical Patchmatch, but also aims to improve the performance with the power of deep learning.

Contributions: **(i)** We introduce the Patchmatch idea into an end-to-end trainable deep learning based MVS framework. Going one step further, we embed the model into a coarse-to-fine framework to speed up computation. **(ii)** We augment the traditional propagation and cost evaluation steps of Patchmatch with learnable, adaptive modules that improve accuracy and base both steps on deep features. We estimate visibility information during cost aggregation for the source views. Moreover, we propose a robust training strategy to introduce randomness into training for improved robustness in visibility estimation and generalization. **(iii)** We verify the effectiveness of our method on various MVS datasets, *e.g.* DTU [1], Tanks & Temples [24] and ETH3D [32]. The results demonstrate that our PatchmatchNet achieves competitive performance, while reducing memory consumption and run-time compared to most learning-based methods.

2. Related Work

Traditional MVS. Traditional MVS methods can be divided into four categories: voxel-based [34, 36], surface evolution based [14,25], patch-based [15,27] and depth map based [16,31,38]. Comparatively, depth map based methods are more concise and flexible. Here, we discuss Patchmatch Stereo methods [16, 31, 38] in this category. Galliani *et al.* [16] present Gipuma, a massively parallel multi-view extension of Patchmatch stereo. It uses a red-black checkerboard pattern to parallelize message-passing during propagation. Schönberger *et al.* [31] present COLMAP, which jointly estimates pixel-wise view selection, depth map and surface normal. ACMM [38] adopts adaptive checkerboard sampling, multi-hypothesis joint view selection and multi-scale geometric consistency guidance. Based on the idea of Patchmatch, we propose our learning-based Patchmatch, which inherits the efficiency from classical Patchmatch, but also improves the performance leveraging deep learning.

Learning-based stereo. GCNet [22] introduces 3D cost volume regularization for stereo estimation and regresses the final disparity map with a soft argmin operation. PSMNet [5] adds spatial pyramid pooling (SPP) and applies a 3D hour-glass network for regularization. DeepPruner [11] develops a differentiable Patchmatch module, without learnable parameters, to discard most disparities and then builds a lightweight cost volume, which is regularized by a 3D CNN. In contrast, we do not apply any cost volume regularization but extend the original Patchmatch idea into the deep learning era. Xu *et al.* [37] propose a sparse point based intra-scale cost aggregation method with deformable convolution [10]. Likewise, we propose a strategy to adaptively sample points for spatial cost aggregation.

Learning-based MVS. Voxel-based methods [20, 21] are restricted to small-scale reconstructions, due to the drawbacks of a volumetric representation. In contrast, based on plane-sweep stereo [9], many recent works [6,28,39,42] use depth maps to reconstruct the scene. They build cost volumes with warped features from multiple views, regularize them with 3D CNNs and regress the depth. As 3D CNNs are time and memory consuming, they usually use downsampled cost volumes. To reduce memory, R-MVSNet [43] sequentially regularizes 2D cost maps with GRU [8] but sacrifices run-time. Current research targets to improve efficiency and also estimate high-resolution depth maps. CasMVSNet [17] proposes cascade cost volumes based on a feature pyramid and estimates the depth map in a coarse-to-fine manner. UCS-Net [7] proposes cascade adaptive thin volumes, which use variance-based uncertainty estimates for an adaptive construction. CVP-MVSNet [41] forms an image pyramid and also constructs a cost volume pyramid. To accelerate propagation in Patchmatch, we likewise employ a cascade formulation. In addition to cascade cost volumes, PVSNet [40] learns to predict visibility for each source image. An anti-noise training strategy is used to introduce disturbing views. We also learn a strategy to adaptively combine the information of multiple views based on visibility information. Moreover, we propose a robust training strategy to include randomness into the training to improve robustness in visibility estimation and generalization. Fast-MVSNet [44] constructs a sparse cost volume to learn a sparse depth map and then use high-resolution RGB image and 2D CNN to densify it. We build a refinement module and use the RGB image to guide the up-sampling of the depth map based on MSG-Net [19].

3. Method

In this section, we introduce the structure of PatchmatchNet, illustrated in Fig. 2. It consists of multi-scale feature extraction, learning-based Patchmatch included iteratively in a coarse-to-fine framework, and a spatial refinement module.

3.1. Multi-scale Feature Extraction

Given N input images of size $W \times H$, we use $\mathbf{I}_0$ and $\{\mathbf{I}_i\}_{i=1}^{N-1}$ to denote reference and source images respectively. Before we apply our learning-based Patchmatch algorithm, we extract pixel-wise features from our inputs,

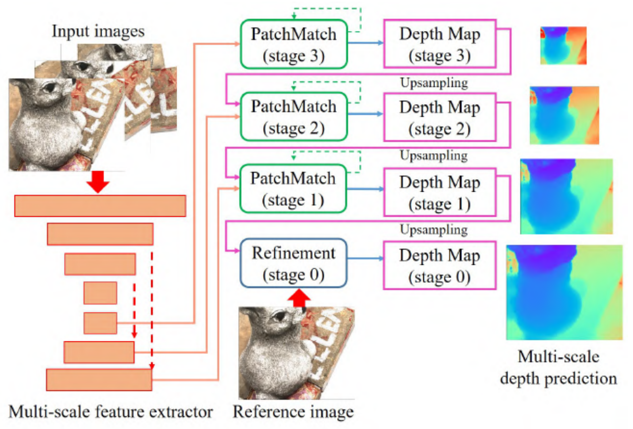

Figure 2: Structure of PatchmatchNet: multi-scale feature extractor, learning-based Patchmatch and refinement. Patchmatch is applied for multiple iterations on several stages to predict the depth map in a coarse-to-fine manner. Refinement uses the input to guide upsampling of the final depth map. On stage k, the resolution of the depth maps is $\frac{W}{2^k} \times \frac{H}{2^k}$, with input images of size $W \times H$.

similar to Feature Pyramid Network (FPN) [26]. Features are extracted hierarchically at multiple resolutions, which allows us to advance our depth map estimation in a coarse-to-fine manner.

3.2. Learning-based Patchmatch

Following traditional Patchmatch [2] and subsequent adaptations to depth map estimation [3, 16], our learnable Patchmatch consists of the following three main steps:

1. Initialization: generate random hypotheses;
2. Propagation: propagate hypotheses to neighbors;
3. Evaluation: compute the matching costs for all the hypotheses and choose best solutions.

After initialization, the approach iterates between propagation and evaluation until convergence. Leveraging deep learning, we propose an adaptive version of the propagation (*Sec.* 3.2.2) and evaluation (*Sec.* 3.2.3) module and also adjust the initialization (*Sec.* 3.2.1). The detailed structure of our Patchmatch pipeline is illustrated in Fig. 3. In a nutshell, the propagation module adaptively samples the points for propagation based on the extracted deep features. Our adaptive evaluation learns to estimate visibility information for cost computation and adaptively samples the spatial neighbors to aggregate the costs again based on deep features. Unlike [3, 16, 38], we refrain from parameterizing the per-pixel hypothesis as a slanted plane, due to heavy memory penalties. Instead, we rely on our learned adaptive evaluation to organize the spatial pattern within the window over which matching costs are computed.

3.2.1 Initialization and Local Perturbation

In the very first iteration of Patchmatch, the initialization is performed in a random manner to promote diversification. Based on a pre-defined depth range $[d_{min}, d_{max}]$, we sample per pixel D_f depth hypotheses in the *inverse* depth range, corresponding to uniform sampling in image space. This helps our model be applicable to complex and large-scale scenes [39, 43]. To ensure we cover the depth range evenly, we divide the (inverse) range into D_f intervals and ensure that each interval is covered by one hypothesis.

For subsequent iterations on stage k, we perform local perturbation by generating per pixel N_k hypotheses uniformly in the normalized inverse depth range R_k and gradually decrease R_k for finer stages. To define the center of R_k, we utilize the estimation from previous iteration, possibly up-sampled from a coarser stage. This delivers a more diverse set of hypotheses than just using propagation. Sampling around the previous estimation can refine the result locally and correct wrong estimates (see supplementary).

3.2.2 Adaptive Propagation

Spatial coherence of depth values does in general only exist for pixel from the same physical surface. Hence, instead of propagating depth hypotheses naively from a static set of neighbors as done for Gipuma [16] and DeepPruner [11], we want to perform the propagation in an adaptive manner, which gathers hypotheses from the same surface. This helps Patchmatch converge faster and deliver more accurate depth maps. Fig. 4 illustrates the idea and functionality of our strategy. Our adaptive scheme tends to gather hypotheses from pixels of the same surface – for both the textured object and the textureless region – enabling us to effectively collect more promising depth hypotheses compared to using just a static pattern.

We base our implementation of the adaptive propagation on Deformable Convolution Networks [10]. As the approach is identical for each resolution stage, we omit subindices denoting the stage. To gather K_p depth hypotheses for pixel $\mathbf{p}$ in the reference image, our model learns additional 2D offsets $\{\Delta\mathbf{o}_i(\mathbf{p})\}_{i=1}^{K_p}$ that are applied on top of fixed 2D offsets $\{\mathbf{o}_i\}_{i=1}^{K_p}$, organized as a grid. We apply a 2D CNN on the reference feature map $\mathbf{F}_0$ to learn additional 2D offsets for each pixel $\mathbf{p}$ and get the depth hypotheses $\mathbf{D}_p(\mathbf{p})$ via bilinear interpolation as follows:

$$\mathbf{D}_p(\mathbf{p}) = \{\mathbf{D}(\mathbf{p} + \mathbf{o}_i + \Delta\mathbf{o}_i(\mathbf{p}))\}_{i=1}^{K_p}, \quad (1)$$

where $\mathbf{D}$ is the depth map from previous iteration, possibly up-sampled from a coarser stage.

3.2.3 Adaptive Evaluation

The adaptive evaluation module performs the following steps: differentiable warping, matching cost computation,

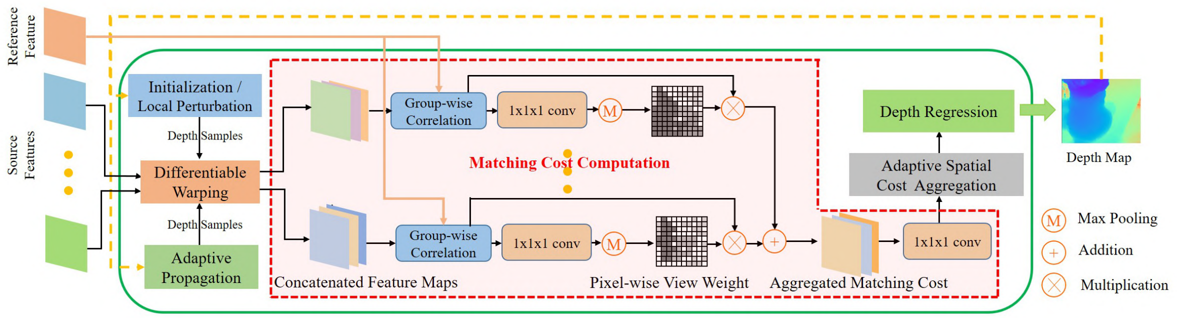

Figure 3: Detailed structure of learned Patchmatch. At the initial iteration of coarsest stage 3 only random depth hypotheses in *initialization* are used. Afterwards, hypotheses are obtained from *adaptive propagation* and *local perturbation*, the latter providing depth samples around the previous estimate. The learned pixel-wise view weight is estimated in the first iteration of Patchmatch and kept fixed in the matching cost computation.

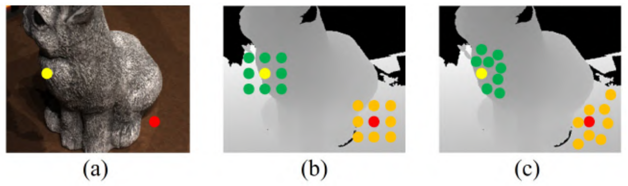

Figure 4: Sampled locations with adaptive propagation. Pixels located at the object boundary (yellow) and a textureless region (red) receive depth hypotheses from sampled neighbors (green and orange). (a) Reference image. (b) Fixed sampling locations of classic propagation. (c) Adaptive sampling locations with adaptive propagation. The grayscale image in (b) and (c) is the ground truth depth map.

adaptive spatial cost aggregation and depth regression. As the approach is identical at each resolution stage, we omit subindices to ease notation.

Differentiable Warping. Following plane sweep stereo [9], most learning-based MVS methods [7, 17, 28, 42, 43] establish front-to-parallel planes at sampled depth hypotheses and warp the feature maps of source images into them. Equipped with intrinsic matrices $\{K_i\}_{i=0}^{K}$ and relative transformations $\{[\mathbf{R}_{0,i}|\mathbf{t}_{0,i}]\}_{i=1}^{K}$ of reference view 0 and source view i, we compute the corresponding pixel $\mathbf{p}_{i,j} := \mathbf{p}_i(d_j)$ in the source for a pixel $\mathbf{p}$ in the reference, given in homogeneous coordinates, and depth hypothesis $d_j := d_j(\mathbf{p})$ as follows:

$$\mathbf{p}_{i,j} = \mathbf{K}_i \cdot (\mathbf{R}_{0,i} \cdot (\mathbf{K}_0^{-1} \cdot \mathbf{p} \cdot d_j) + \mathbf{t}_{0,i}). \tag{2}$$

We obtain the warped source feature maps of view i and the j-th set of (per pixel different) depth hypotheses, $\mathbf{F}_i(\mathbf{p}_{i,j})$, via differentiable bilinear interpolation.

Matching Cost Computation. For multi-view stereo, this step has to integrate information from an arbitrary number of source views into a single cost per pixel $\mathbf{p}$ and depth hypothesis d_j. To that end, we compute the cost per hypothesis via group-wise correlation [39] and aggregate over the views with a pixel-wise view weight [31, 38, 40]. In that manner we can employ visibility information during cost aggregation and gain robustness. Finally, the per group costs are projected into a single number, per reference pixel and hypothesis, by a small network.

Let $\mathbf{F}_0(\mathbf{p})$, $\mathbf{F}_i(\mathbf{p}_{i,j}) \in \mathbb{R}^C$ be the features in the reference and source feature maps respectively. After dividing their feature channels evenly into G groups, $\mathbf{F}_0(\mathbf{p})^g$ and $\mathbf{F}_i(\mathbf{p}_{i,j})^g$, the g-th group similarity $\mathbf{S}_i(\mathbf{p},j)^g \in \mathbb{R}$ is computed as:

$$\mathbf{S}_i(\mathbf{p},j)^g = \frac{G}{C} \langle \mathbf{F}_0(\mathbf{p})^g, \mathbf{F}_i(\mathbf{p}_{i,j})^g \rangle \,, \tag{3}$$

where $\langle \cdot,\cdot \rangle$ is the inner product. We use $\mathbf{S}_i(\mathbf{p},j) \in \mathbb{R}^G$ to denote the respective group similarity vector. Agglomeration over hypotheses and pixels delivers the tensor $\mathbf{S}_i \in \mathbb{R}^{W\times H\times D\times G}$.

To find pixel-wise view weights, $\{\mathbf{w}_i(\mathbf{p})\}_{i=1}^{N-1}$, we exploit the diversity of our initial set of depth hypotheses in the first iteration on stage 3 (*Sec.* 3.2.1). We intend $\mathbf{w}_i(\mathbf{p})$ to represent the visibility information of pixel $\mathbf{p}$ in the source image $\mathbf{I}_i$. The weights are computed once and kept fixed and up-sampled for finer stages.

A simple pixel-wise view weight network, composed of 3D convolution layers with 1×1×1 kernels and sigmoid nonlinearities, takes the initial set of similarities $\mathbf{S}_i$ to output a number between 0 and 1 per pixel and depth hypothesis to produce $\mathbf{P}_i \in \mathbb{R}^{W\times H\times D}$. The view weights for pixel $\mathbf{p}$ and source image $\mathbf{I}_i$ are given by:

$$\mathbf{w}_i(\mathbf{p}) = \max\{\mathbf{P}_i(\mathbf{p},j) | j = 0,1,\ldots,D-1\}\,, \tag{4}$$

where $\mathbf{P}_i(\mathbf{p},j)$ intuitively represents confidence of visibility for the range covered by the j-th depth hypothesis at $\mathbf{p}$.

The final per group similarities $\bar{\mathbf{S}}(\mathbf{p},j)$ for pixel $\mathbf{p}$ and the j-th hypothesis are the weighted sum of $\mathbf{S}_i(\mathbf{p},j)$ and the view weight $\mathbf{w}_i(\mathbf{p})$:

$$\bar{\mathbf{S}}(\mathbf{p},j) = \frac{\sum_{i=1}^{N-1} \mathbf{w}_i(\mathbf{p}) \cdot \mathbf{S}_i(\mathbf{p},j)}{\sum_{i=1}^{N-1} \mathbf{w}_i(\mathbf{p})}. \tag{5}$$

Finally, we compose $\bar{\mathbf{S}}(\mathbf{p},j)$ for all pixels and hypothe-

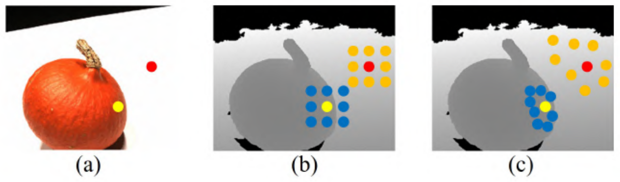

Figure 5: Sampled locations in adaptive spatial cost aggregation. Pixels located at an object boundary (yellow) and a textureless area (red) aggregate matching costs from sampled neighbors (blue and orange). (a) Reference image for depth prediction. (b) Fixed sampling locations. (c) Adaptive sampling locations of our method. The grayscale image in (b) and (c) is the ground truth depth map.

ses into $\bar{\mathbf{S}} \in \mathbb{R}^{W \times H \times D \times G}$ and apply a small network with 3D convolution and $1 \times 1 \times 1$ kernels to obtain a single cost, $\mathbf{C} \in \mathbb{R}^{W \times H \times D}$, per pixel and depth hypothesis.

Adaptive Spatial Cost Aggregation. Traditional MVS matching algorithms often aggregate costs over a spatial window (*i.e.* in our case a front-to-parallel plane) for increased matching robustness and an implicit smoothing effect. Arguably, our multi-scale feature extractor already aggregates neighboring information from a large receptive field in the spatial domain. Nevertheless, we propose to look into spatial cost aggregation. To prevent the problem of aggregating across surface boundaries, we propose an adaptive spatial aggregation strategy based on Patchmatch stereo [3] and AANet [37]. For a spatial window of K_e pixels $\{\mathbf{p}_k\}_{k=1}^{K_e}$ are organized as a grid, we learn per pixel additional offsets $\{\Delta\mathbf{p}_k\}_{k=1}^{K_e}$. The aggregated spatial cost $\tilde{\mathbf{C}}(\mathbf{p}, j)$ is defined as:

$$\tilde{\mathbf{C}}(\mathbf{p}, j) = \frac{1}{\sum_{k=1}^{K_e} w_k d_k} \sum_{k=1}^{K_e} w_k d_k \mathbf{C}(\mathbf{p}+\mathbf{p}_k+\Delta\mathbf{p}_k, j), \quad (6)$$

where w_k and d_k weight the cost $\mathbf{C}$ based on feature and depth similarity (details in supplementary). Similar to adaptive propagation, the per pixel sets of displacements $\{\Delta\mathbf{p}_k\}_{k=1}^{K_e}$ are found by applying a 2D CNN on the reference feature map $\mathbf{F}_0$. Fig. 5 exemplifies the learned adaptive aggregation window. Sampled locations stay within object boundaries, while for the textureless region, the sampling points aggregate over a larger spatial context, which can potentially reduce the ambiguity of estimation.

Depth Regression. Using *softmax* we turn the (negative) cost $\tilde{\mathbf{C}}$ to a probability $\mathbf{P}$, which is used for sub-pixel depth regression and measure estimation confidence [42]. The regressed depth value $\mathbf{D}(\mathbf{p})$ at pixel $\mathbf{p}$ is found as the expectation w.r.t. $\mathbf{P}$ of the hypotheses:

$$\mathbf{D}(\mathbf{p}) = \sum_{j=0}^{D-1} d_j \cdot \mathbf{P}(\mathbf{p}, j). \quad (7)$$

3.3. Depth Map Refinement

Instead of using Patchmatch also on the finest resolution level (stage 0), we find it sufficient to directly up-sample (from resolution $\frac{W}{2} \times \frac{H}{2}$ to $W \times H$) and refine our estimation with the RGB image. Based on MSG-Net [19], we design a depth residual network. To avoid being biased for a certain depth scale, we pre-scale the input depth map into the range $[0, 1]$ and convert it back after refinement. Our refinement network learns to output a residual that is added to the (up-sampled) estimation from Patchmatch, $\mathbf{D}$, to get the refined depth map $\mathbf{D}_{ref}$. This network independently extracts feature maps $\mathbf{F}_D$ and $\mathbf{F}_I$ from $\mathbf{D}$ and $\mathbf{I}_0$ and applies deconvolution on $\mathbf{F}_D$ to up-sample the feature map to the image size. Multiple 2D convolution layers are applied on top of the concatenation of both feature maps – depth map and image – to deliver the depth residual.

3.4. Loss Function

Loss function L_{total} considers the losses among all the depth estimation and rendered ground truth with same resolution as a sum:

$$L_{total} = \sum_{k=1}^{3} \sum_{i=1}^{n_k} L_i^k + L_{ref}^0. \quad (8)$$

We adopt the smooth $L1$ loss for L_i^k, the loss of the i-th iteration of Patchmatch on stage k ($k = 1, 2, 3$) and L_{ref}^0, the loss for final refined depth map.

4. Experiments

We evaluate our work on multiple datasets, such as DTU [1], Tanks & Temples [24] and ETH3D [32] and analyze each new component with an ablation study.

4.1. Datasets

The DTU dataset [1] is an indoor multi-view stereo dataset with 124 different scenes where all scenes share the same camera trajectory. We use the training, testing and validation split introduced in [20]. The Tanks & Temples dataset [24] is provided as a set of video sequences in realistic environments. It is divided into intermediate and advanced datasets. ETH3D benchmark [32] consists of calibrated high-resolution images of scenes with strong viewpoint variations. It is divided into training and test datasets.

4.2. Robust Training Strategy

Many learning-based methods [7,17,28,39,41,42] select two best source views based on view selection scores [42] to train models on DTU [1]. However, the selected source views have a strong visibility correlation with the reference view, which may affect the training of the pixel-wise view weight network. Instead, we propose a robust training strategy based on PVSNet [40]. For each reference view, we

Methods	Acc.(mm)	Comp.(mm)	Overall(mm)
Camp [4]	0.835	0.554	0.695
Furu [15]	0.613	0.941	0.777
Tola [35]	0.342	1.190	0.766
Gipuma [16]	**0.283**	0.873	0.578
SurfaceNet [20]	0.450	1.040	0.745
MVSNet [42]	0.396	0.527	0.462
R-MVSNet [43]	0.383	0.452	0.417
CIDER [39]	0.417	0.437	0.427
P-MVSNet [28]	0.406	0.434	0.420
Point-MVSNet [6]	0.342	0.411	0.376
Fast-MVSNet [44]	0.336	0.403	0.370
CasMVSNet [17]	0.325	0.385	0.355
UCS-Net [7]	0.338	0.349	**0.344**
CVP-MVSNet [41]	0.296	0.406	0.351
Ours	0.427	**0.277**	0.352

Table 1: Quantitative results of different methods on DTU's evaluation set [1] (lower is better).

randomly choose four from the ten best source views for training. This strategy increases the diversity at training time and augments the dataset on the fly, which improves the generalization performance. In addition, training on those random source views with weak visibility correlation generates further robustness for our visibility estimation.

4.3. Implementation Details

We implement the model with PyTorch [30] and train it on DTU's training set [1]. We set the image resolution to 640×512 and the number of input images to $N = 5$. The selection of source images is based on the proposed robust training strategy. We set the iteration number of Patchmatch on stages $3, 2, 1$ as $2, 2, 1$. For initialization, we set $D_f = 48$. For local perturbation, we set $R_3 = 0.38, R_2 = 0.09, R_1 = 0.04$ (see supplementary), $N_3 = 16, N_2 = 8, N_1 = 8$. In the adaptive propagation, we set K_p on stages $3, 2, 1$ to 16, 8, 0 (no propagation for last iteration on stage 1, see supplementary). For the adaptive evaluation, we use $K_e = 9$ on all stages. We train our model with Adam [23] ($\beta_1 = 0.9, \beta_2 = 0.999$) for 8 epochs with a learning rate of 0.001. Here, we use a batch size of 4 and train on 2 Nvidia GTX 1080Ti GPUs. After depth estimation, we reconstruct point clouds similar to MVSNet [42].

4.4. Benchmark Performance

Evaluation on DTU Dataset. We input images at their original size (1600×1200) and set the number of views N to 5. The depth range for sampling depth hypotheses is fixed to $[425\,mm, 935\,mm]$. We follow the evaluation metrics provided by the DTU dataset [1]. As shown in Table 1, while Gipuma [16] performs best in *accuracy*, our method outperforms others in *completeness* and achieves competitive performance in *overall quality*. Fig. 6 shows qualitative results. Our solution reconstructs a denser point cloud with finer details, which reflects in a high completeness. Further,

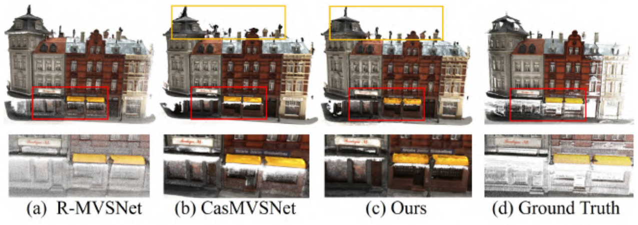

Figure 6: Qualitative comparison of scan 9 of DTU [1]. *Top*: Reconstructed and ground truth point clouds. Our method preserves the thin structures on the roof better than CasMVSNet [17] and delivers accurate boundaries. *Bottom*: Zoom in. Capable of handling a high input resolution, our result is much denser, with finer details at doors, windows and logos.

our reconstruction at boundaries and thin structures appears better than of CasMVSNet [17]. Our adaptive propagation can recover from errors at boundaries, induced at coarser resolutions, by using the information of neighbors inside the boundary (*c.f.* Fig. 4), while solely relying on sampling around a previous estimation, as CasMVSNet, can fail.

Memory and Run-time Comparison. We compare the memory consumption and run-time with several state-of-the-art learning-based methods that achieve competing performance with low memory consumption and run-time: CasMVSNet [17], UCS-Net [7] and CVP-MVSNet [41]. These methods propose a cascade formulation of 3D cost volumes and output depth maps at the same resolution as the input images. As shown in Fig. 7, memory and run-time of all the methods increase almost linearly w.r.t. the resolution as the number of depth hypotheses is fixed (notably this will lead to under-sampling of the enlarged image space for methods using a naive cost volume approach). Note that at higher resolutions other methods could not fit into the memory of the GPU used for evaluation. We further observe that memory consumption and run-time increase much slower for PatchmatchNet than for other methods. For example, at a resolution of 1152×864 (51.8%), memory consumption and run-time are reduced by 67.1% and 66.9% compared to CasMVSNet, by 55.8% and 63.9% compared to UCS-Net and by 68.5% and 83.4% compared to CVP-MVSNet. Combining the results in Table 1, we conclude that our method is much more efficient in memory consumption and run-time than most state-of-the-art learning-based methods, at a very competitive performance.

Evaluation on Tanks & Temples Dataset. We use the model trained on DTU [1] without any fine-tuning. For evaluation, we set the input image size to 1920×1056 and the number of views N to 7. The camera parameters and sparse point cloud are recovered with OpenMVG [29]. During evaluation, the GPU memory and run-time for each depth map are 2887 MB and 0.505 s respectively. As shown in Table 2, the performance of our method on the interme-

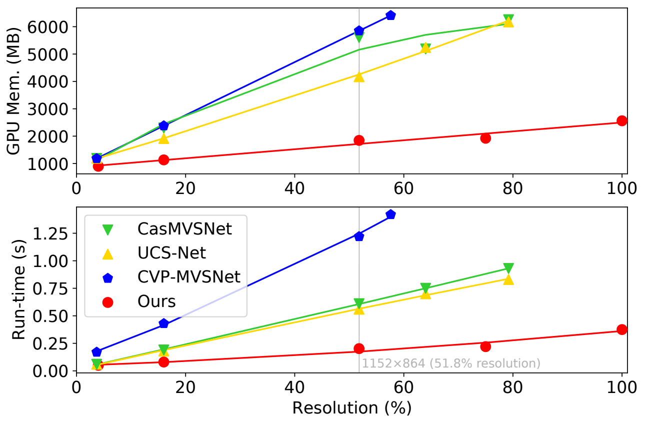

Figure 7: Relating GPU memory and run-time to the input resolution on DTU's evaluation set [1]. The original image resolution is 1600×1200 (100%). Note that at higher resolutions other methods could not fit into the memory of a Nvidia RTX 2080 GPU, which is used for evaluation.

Methods	Intermediate	Advanced
COLMAP [31]	42.14	27.24
MVSNet [42]	43.48	-
R-MVSNet [43]	48.40	24.91
CIDER [39]	46.76	23.12
P-MVSNet [28]	55.62	-
Point-MVSNet [6]	48.27	-
Fast-MVSNet [44]	47.39	-
CasMVSNet [17]	**56.42**	31.12
UCS-Net [7]	54.83	-
CVP-MVSNet [41]	54.03	-
Ours	53.15	**32.31**

Table 2: Results of different methods on Tanks & Temples [24] (F score, higher is better). Note that most methods refrain to evaluate on the more challenging *Advanced* dataset.

diate dataset is comparable to CasMVSNet [17], which has the highest score. For the more complex advanced dataset, our method performs best among all the methods. Overall, due to its simple, scalable structure, our PatchmatchNet demonstrates competitive generalization performance, low memory consumption and low run-time compared to state-of-the-art learning-based methods that commonly use 3D cost volume regularization.

Evaluation on ETH3D Benchmark. We use the model trained on DTU [1] without any fine-tuning. For evaluation, we set the input image size as 2688×1792. Due to the strong viewpoint changes in ETH3D, we also use $N = 7$ views to utilize more multi-view information. Camera parameters and the sparse point cloud are recovered with COLMAP [31]. During evaluation, the GPU memory consumption and run-time for the estimation of each depth map are 5529 MB and 1.250 s respectively. As shown in Table 3, on the training dataset, the performance of our method is comparable to COLMAP [31] and PVSNet [40]. On the particularly challenging test dataset, our method performs

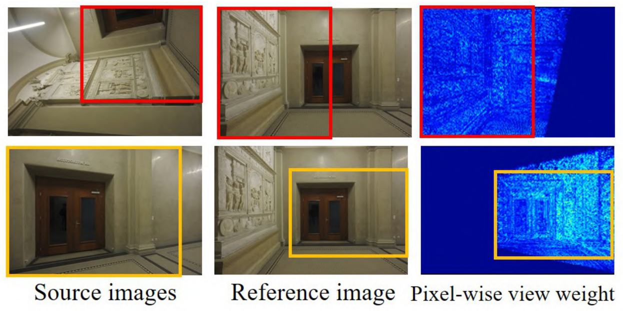

Figure 8: Visualization of our pixel-wise view weight on a scene from ETH3D [32]. Areas marked with boxes in source images and reference image are co-visible. *Right*: The corresponding pixel-wise view weights, bright colors (large values) indicate co-visibility.

Methods	Training		Test	
	F_1 score	Time(s)	F_1 score	Time(s)
MVE [13]	20.47	13278.69	30.37	10550.67
Gipuma [16]	36.38	587.77	45.18	689.75
PMVS [15]	46.06	836.66	44.16	957.08
COLMAP [31]	**67.66**	2690.62	73.01	1658.33
PVSNet [40]	67.48	-	72.08	829.56
Ours	64.21	**452.63**	**73.12**	**492.52**

Table 3: Results of different methods on ETH3D [32] (F_1 score, higher is better). Due to strong viewpoint variations, currently, the only competitive pure learning-based method submitted on ETH3D is PVSNet [40].

best among all methods. Furthermore, our method is the fastest one evaluated so far (November 16th, 2020) on the ETH3D benchmark. Noting that PVSNet is a state-of-the-art learning-based method, the quantitative results demonstrate the effectiveness, efficiency and generalization capabilities of our method.

We visualize the pixel-wise view weight in Fig. 8. Brighter colors indicate co-visible areas, *i.e*. regions in the reference image that are also (well) visible in the source images. Conversely, areas that are not visible in the source images receive a dark color, corresponding to a low weight. Also pixel near depth discontinuities appear slightly darker than surrounding areas. By inspection, we conclude that our pixel-wise view weight is indeed capable to determine co-visible areas between the reference and source views.

4.5. Ablation Study

We conduct an ablation study to analyze the components. Unless specified, all the following studies are done on DTU's evaluation set [1].

Adaptive Propagation (AP) & Adaptive Evaluation (AE). We compare our base model with versions that employ fixed 2D offsets, similar to Gipuma [16], for propagation (w/o AP), and, fixed 2D offsets to sample the neighbors for spatial cost aggregation in the evaluation step (w/o AE). As shown in Table 4, our adaptive propagation and adaptive evaluation modules each improve results w.r.t. accuracy as

Methods	Acc.(mm)	Comp.(mm)	Overall(mm)
w/o AP & AE	0.453	0.339	0.396
w/o AP	0.437	0.285	0.361
w/o AE	0.437	0.324	0.380
Ours	**0.427**	**0.277**	**0.352**

Table 4: Parameter sensitivity on DTU [1] for Adaptive Propagation (AP) and Adaptive Evaluation (AE).

Iterations	Acc.(mm)	Comp.(mm)	Overall(mm)
1,1,1	0.446	0.278	0.362
2,2,1	0.427	**0.277**	0.352
3,3,1	**0.425**	0.277	**0.351**
4,4,1	0.425	0.277	0.351
5,5,1	0.425	0.277	0.351

Table 5: Ablation study of the number of Patchmatch iterations on DTU [1]. *Iterations*, *'a,b,c'* means that there are a, b and c iterations on stage 3, 2 and 1. Thanks to the coarse-to-fine framework, learned adaptive propagation, diverse initialization and local perturbation, our method converges after only 5 iterations combined on all stages.

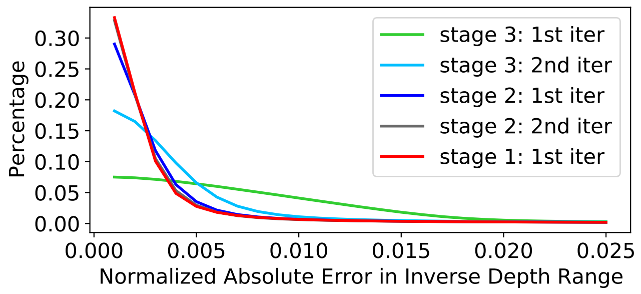

Figure 9: Distribution of normalized absolute errors in the inverse depth range on DTU's evaluation set [1]. '*Stage k*, n^{th} *iter*' denotes the result of the n^{th} iteration of Patchmatch on stage k. Already after stage 3 our estimate is close to the solution while stage 2 and 1 refine it even more.

well as completeness.

Number of Iterations of Patchmatch. Recall that, during training (*Sec.* 4.3), we do not include adaptive propagation for Patchmatch on stage 1. Consequently, we also keep the number of iterations on stage 1 as 1 for this investigation. More iterations of Patchmatch generally improve the performance, yet, the improvements stagnate after '2,2,1' iterations, Table 5. We further visualize the distribution of normalized absolute error in the inverse depth range for the setting '2,2,1' in Fig. 9. We observe that the error converges after only 5 iterations of Patchmatch across all stages. Compared to Gipuma [16] that employs a large number of neighbors for propagation, we have embedded Patchmatch in a coarse-to-fine framework to speed up long range interactions. Apart from that, our learned adaptive propagation, diverse initialization and local perturbation all contribute towards a faster converge.

Pixel-wise View Weight (VW) & Robust Training Strategy (RT). In this experiment, we resign from pixel-wise view weighting (w/o VW) and do not follow our strategy but choose four best source views for training (w/o RT). To investigate the generalization performance, we further test on Tanks & Temples [24] and ETH3D [32]. Table 6 shows a similar performance on DTU [1] for all the models, yet, we observe a drop in performance on the other datasets without pixel-wise view weight or the robust training strategy. This proves that these two modules lead to improved robustness and a better generalization performance.

Methods	DTU	Tanks&Temples Intermediate	ETH3D Training
w/o VW & RT	0.351	52.05	60.40
w/o VW	0.353	52.46	61.32
w/o RT	**0.348**	52.12	62.57
Ours	0.352	**53.15**	**64.21**

Table 6: Ablation study concerning the pixel-wise view weight (VW) and the robust training strategy (RT).

N	Acc.(mm)	Comp.(mm)	Overall(mm)
2	0.439	0.332	0.385
3	0.428	0.284	0.356
5	**0.427**	**0.277**	**0.352**
6	0.429	0.278	0.353

Table 7: Ablation study of the number of input views N on DTU's evaluation set [1].

Number of Views. Apart form our standard setting of $N = 5$ views for DTU's evaluation set [1], we also evaluate the performance when $N = 2, 3, 6$. Using more views is known to improve performance, *e.g.* by alleviating the occlusion problem, which coincides with our findings summarized in Table 7. With more input views, the reconstruction quality tends to improve in both accuracy and completeness.

5. Conclusion

We present PatchmatchNet, a novel cascade formulation of learning-based Patchmatch, augmented with learned adaptive propagation and evaluation modules based on deep features. Inherited from its name-giving ancestor, PatchmatchNet naturally possesses low memory requirements, independent of the disparity range and unlike most learning-based methods, PatchmatchNet does not rely on 3D cost volume regularization. Embedded into a cascade formulation, PatchmatchNet further shows a high processing speed. Despite its simple structure, extensive experiments on DTU, Tanks & Temples and ETH3D demonstrate a remarkably low computation time, low memory consumption, favorable generalization properties and competitive performance compared to the state-of-the-art. PatchmatchNet makes learning-based MVS more efficient and more applicable to memory restricted devices or time critical applications. For the future, we hope to apply it on movable platforms such as mobile phones and head mounted displays, where the computation resource is limited.

Supplementary Material

1. Why not use 3D cost volume regularization?

The adaptive evaluation of our learning-based Patchmatch utilizes 3D convolution layers with $1 \times 1 \times 1$ kernels for the matching cost computation as well as the pixel-wise view weight estimation. This is in contrast to common previous works [7, 17, 28, 39, 40, 41, 42] where a 3D U-Net regularizes the cost volume. Similarly, arguing that the distribution of cost volume itself being not discriminative enough [12, 33], PVSNet [40] also applies a 3D U-Net for predicting the visibility per source view.

The problem with such regularization framework is that it requires a regular spatial structure in the volume. Although we concatenate the matching costs per pixel and depth hypothesis into a volume-like shape as other works [7, 17, 28, 39, 40, 41, 42], we do not possess such a regular structure: (i) the depth hypotheses for each pixel and its spatial neighbors are different, which makes it difficult to aggregate cost information in the spatial domain; (ii) the depth hypotheses of each pixel are not uniformly distributed in the inverse depth range as CIDER [39], which makes it difficult to aggregate cost information along depth dimension.

Recall that during the computation of the pixel-wise view weights in the initial iteration of Patchmatch, depth hypotheses are randomly distributed in the *inverse* depth range, *i.e.* the hypotheses are spatially different per pixel. In each subsequent iteration (on stage k), we perform local perturbation by generating per pixel N_k depth hypotheses uniformly in the normalized inverse depth range R_k, which is centered at the previous estimate. Consequently, the hypotheses of spatial neighbors can differ significantly, especially at depth discontinuities and thin structures. Including the hypotheses obtained by adaptive propagation, that are, moreover, not uniform in the inverse depth range, will increase these effects further.

In the end, however, the main reason for our approach to avoid 3D cost volume regularization altogether is efficiency. In a coarse-to-fine framework, running such regularization frameworks over multiple iterations of Patchmatch on each stage would increase memory consumption and run-time significantly and mitigate our main contribution of building a high-performance, but particularly lightweight framework that can operate with a high computation speed.

2. How to set the normalized inverse depth range R_k in the local perturbation step of Patchmatch?

After the initial iteration, our set of hypothesis is obtained by adaptive propagation and by local perturbation of the previous estimation. Recall that our local perturbation procedure enriches the set of hypothesis by generating per pixel N_k depth hypotheses uniformly in the normalized inverse depth range R_k. The objective is two-fold. Especially at the beginning, at low resolution, this helps to further explore the search space. More importantly, our adaptive propagation implicitly assumes front-to-parallel surfaces, since we do not explicitly include tangential surface information (due to an implied heavy memory consumption) like [3, 16, 38]. Sampling in the local vicinity of the previous estimation will refine the solution locally and mitigate potential disadvantages from not explicitly modeling tangential surface information. We find it helpful to apply these perturbations already at an early stage to inject the positive effects into hypothesis propagation and note that a-posteriori refinement at the finest level alone cannot recover the same quality. In practice, we again operate in coarse-to-fine manner and set R_k accordingly, based on the hierarchy level.

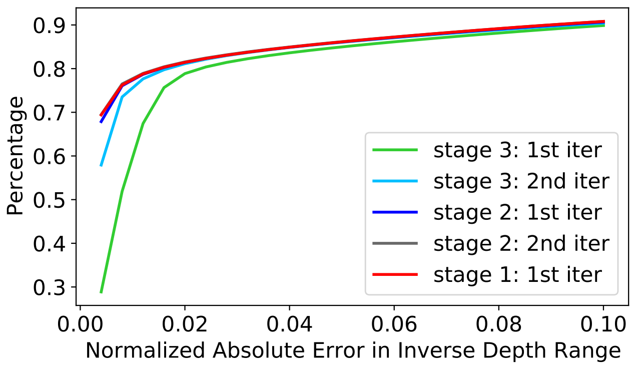

Figure 10: Cumulative distribution function of normalized absolute errors in the inverse depth range on DTU's evaluation set [1]. '*Stage k, n^{th} iter*' denotes the result of the n^{th} iteration of Patchmatch on stage k.

Fig. 10 shows the cumulative distribution function of the normalized absolute error in the inverse depth range on DTU's evaluation set [1]. After the first iteration of Patchmatch on stage 3, the estimation error decreases remarkably: the normalized error is already smaller than 0.1 for 90.0% percent of the cases. Visibly, the performance keeps improving after each iteration. To correct errors in estimation and refine the results on stage k, we set R_k to compensate most of estimation errors. For example, we set $R_3 = 0.38$ for Patchmatch on stage 3 after first iteration so that we can cover most ground truth depth in the hypothesis range and then refine the results. Besides, adaptive propagation will further correct those wrong estimations with the depth hypotheses from neighbors when sampling in R_k fails in refinement (*c.f*. Fig. 6 from the paper).

3. Why not include propagation for last iteration of Patchmatch on stage 1?

Similar to MVSNet [42], the point cloud reconstruction mainly consists of photometric consistency filtering, geometric consistency filtering and depth fusion. Photometric consistency filtering is used to filter out those depth hypotheses that have low confidence. Based on MVSNet [42], we define the confidence as the probability sum of the depth hypotheses that fall in a small range near the estimation. We use the probability $\mathbf{P}$ (*c.f*. Eq. 7 from the paper) from the last iteration of Patchmatch on stage 1 for filtering. In this iteration, we only perform local perturbation, without adaptive propagation. At stage 1, operating at a quarter the image resolution and with the algorithm almost converged, the hypotheses obtained via propagation from spatial neighbors are usually very similar to the current solution. Such irregular sampling of the probability space causes bias in the regression (*c.f*. Eq. 7 from the paper) and the estimate becomes over-confident at the current solution, where most propagated samples are located. In contrast, by performing only the local perturbation, the depth hypotheses are uniformly distributed in the inverse depth range. Contrary to previous iterations, we compute the estimated depth at pixel $\mathbf{p}$, $\mathbf{D}(\mathbf{p})$, by utilizing the inverse depth regression [39], which is based on the *soft argmin* operation [22]:

$$\mathbf{D}(\mathbf{p}) = (\sum_{j=0}^{D-1} \frac{1}{d_j} \cdot \mathbf{P}(\mathbf{p}, j))^{-1}, \tag{9}$$

where $\mathbf{P}(\mathbf{p}, j)$ is the probability for pixel $\mathbf{p}$ at the j-th depth hypothesis. Then we compute the probability sum of four depth hypotheses that are nearest to the estimation to measure the confidence [42].

4. Weighting in the Adaptive Spatial Cost Aggregation

Recall that in Eq. 6 of the paper we utilize two weights to aggregate our spatial costs, $\{w_k\}_{k=1}^{K_e}$ based on spatial feature similarity and $\{d_k\}_{k=1}^{K_e}$ based on the similarity of depth hypotheses. The feature weights $\{w_k\}_{k=1}^{K_e}$ at a pixel $\mathbf{p}$ are based on the feature similarity at the sampling locations around $\mathbf{p}$, measured in the reference feature map $\mathbf{F}_0$. Given the sampling positions $\{\mathbf{p} + \mathbf{p}_k + \Delta\mathbf{p}_k\}_{k=1}^{K_e}$, we extract the corresponding features from $\mathbf{F}_0$ via bilinear interpolation. Then we apply group-wise correlation [18] between the features at each sampling location and $\mathbf{p}$. The results are concatenated into a volume on which we apply 3D convolution layers with $1 \times 1 \times 1$ kernels and sigmoid non-linearities to output normalized weights that describe the similarity between each sampling point and $\mathbf{p}$.

As discussed in *Sec.* 1, neighboring pixels will be assigned different depth values throughout the estimation process. For pixel $\mathbf{p}$ and the j-th depth hypothesis, our depth weights $\{d_k\}_{k=1}^{K_e}$ take this into account and downweight the influence of samples with large depth difference, especially when located across depth discontinuities. To that end, we collect the absolute difference in inverse depth between each sampling point and pixel $\mathbf{p}$ with their j-th hypotheses, and obtain the weights by applying a sigmoid function on the, again, inverted differences for normalization.

Stages	Acc.(mm)	Comp.(mm)	Overall(mm)
3	0.740	0.389	0.564
2	0.471	0.283	0.377
1	0.441	**0.268**	0.354
0	**0.427**	0.277	**0.352**

Table 8: Quantitative results of different stages on DTU's evaluation set [1] (lower is better). The depth maps on stages 3, 2 and 1 are upsampled to reach the same resolution as input images and then used to reconstruct point clouds.

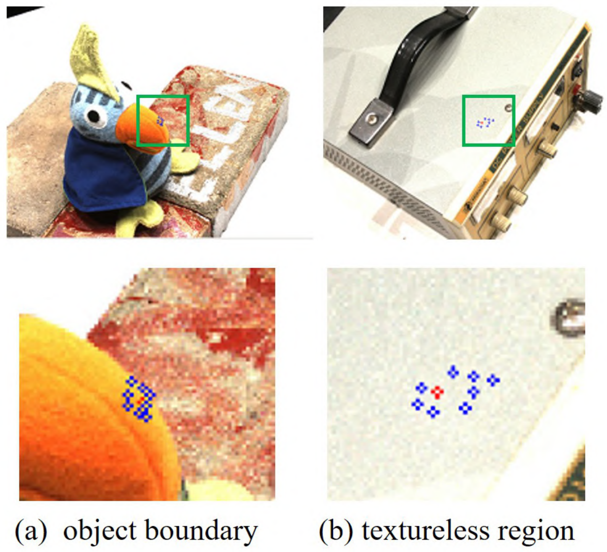

Figure 11: Visualization of sampling locations in adaptive propagation for two typical situations: object boundary and textureless region. The center points and sampling points are shown in red and blue respectively.

5. Evaluation of Multi-stage Depth Estimation

We use multiple stages to estimate the depth map in a coarse-to-fine manner. Here, we analyze the effectiveness of our multi-stage framework. We upsample the estimated depth maps on stages 3, 2 and 1, to the same resolution as the input and then reconstruct the point clouds. As shown in Table 8, the reconstruction quality gradually increases from coarser stages to finer ones. This shows that our multi-stage framework can reconstruct the scene geometry with increasing accuracy and completeness.

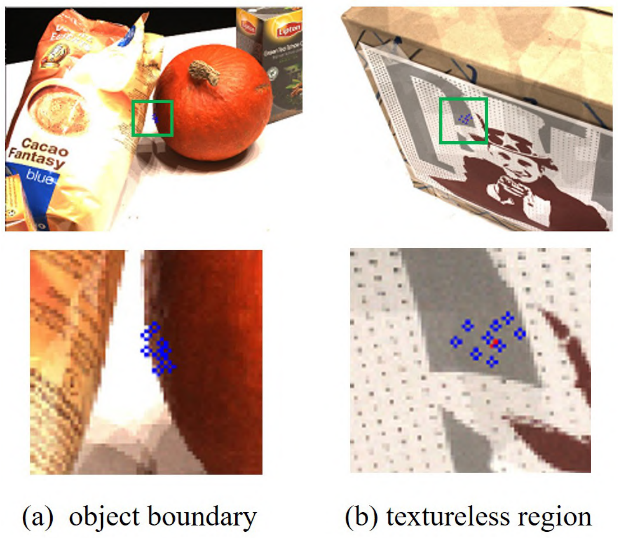

Figure 12: Visualization of sampling locations in adaptive evaluation for two typical situations: object boundary and textureless region. The center points and sampling points are shown in red and blue respectively.

6. Visualization of Adaptive Propagation

We visualize the sampling locations in two typical situations, at an object boundary and a textureless region. As shown in Fig. 11, for the pixel $\mathbf{p}$ at the object boundary, all sampling points tend to be located on the same surface as $\mathbf{p}$. In contrast, for the pixel $\mathbf{q}$ in the textureless region, the sampling points are spread over a larger region. By sampling from a large region, a more diverse set of depth hypotheses can be propagated to $\mathbf{q}$ and reduce the local ambiguity for depth estimation in the textureless area. The visualization shows two examples how the adaptive propagation successfully adapts the sampling to different challenging situations.

7. Visualization of Adaptive Evaluation

Here, we again visualize the sampling locations for two situations, at an object boundary and a textureless region. Fig.12 demonstrates that for the pixel $\mathbf{p}$ at the object boundary, sampling points tend to stay within the boundaries of the object, such that they focus on similar depth regions. For the pixel $\mathbf{q}$ in the textureless region, the points are distributed sparsely to sample from a large context, which helps to obtain reliable matching and to reduce the ambiguity. Again, the visualization demonstrates how our adaptive evaluation adapts the sampling for the spatial cost aggregation to different situations.

8. Visualization of Point Clouds

We visualize reconstructed point clouds from DTU's evaluation set [1], Tanks & Temples dataset [24] and ETH3D benchmark [32] in Fig. 13, 14, 15.

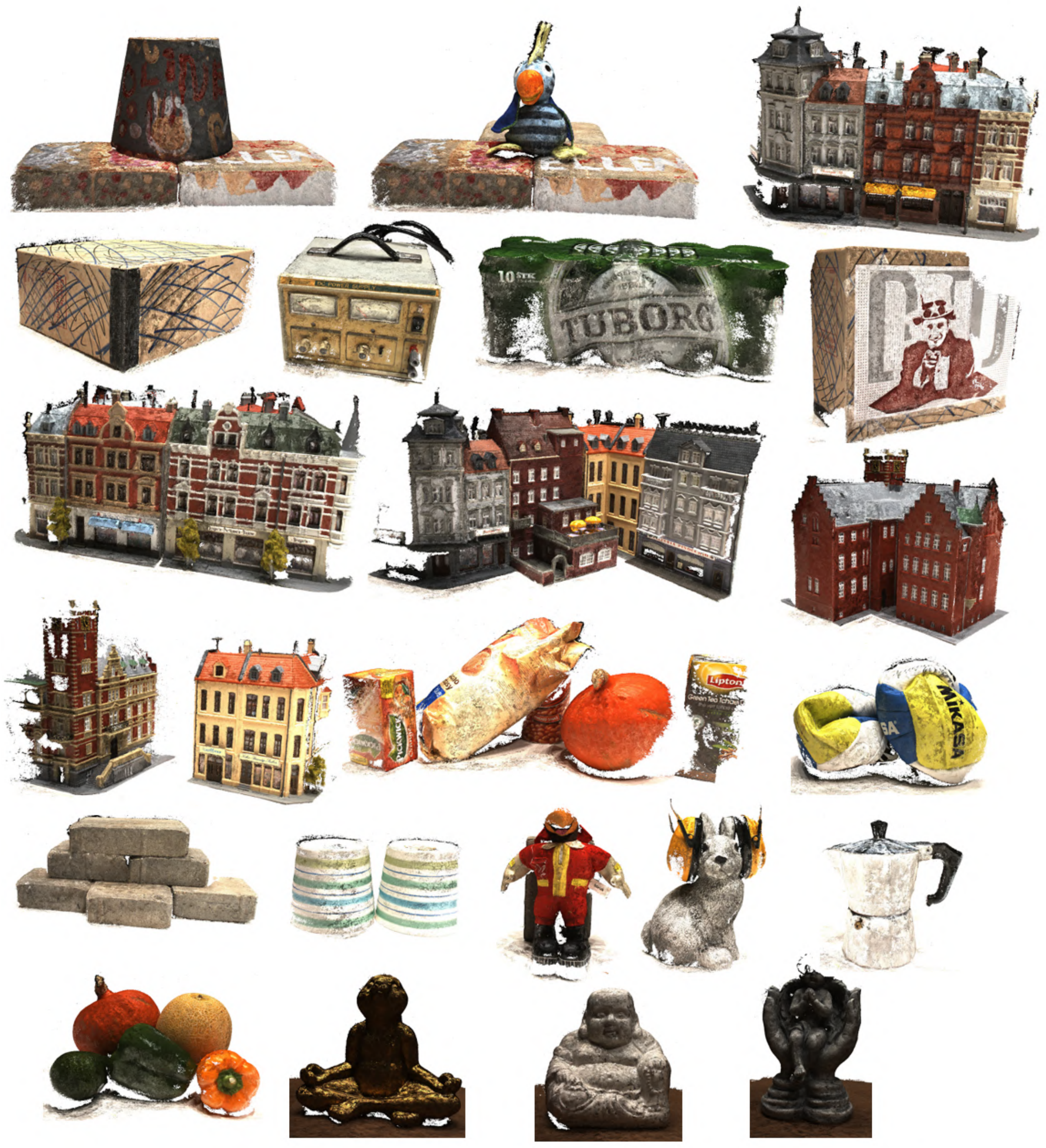

Figure 13: Reconstruction results on DTU's evaluation set [1].

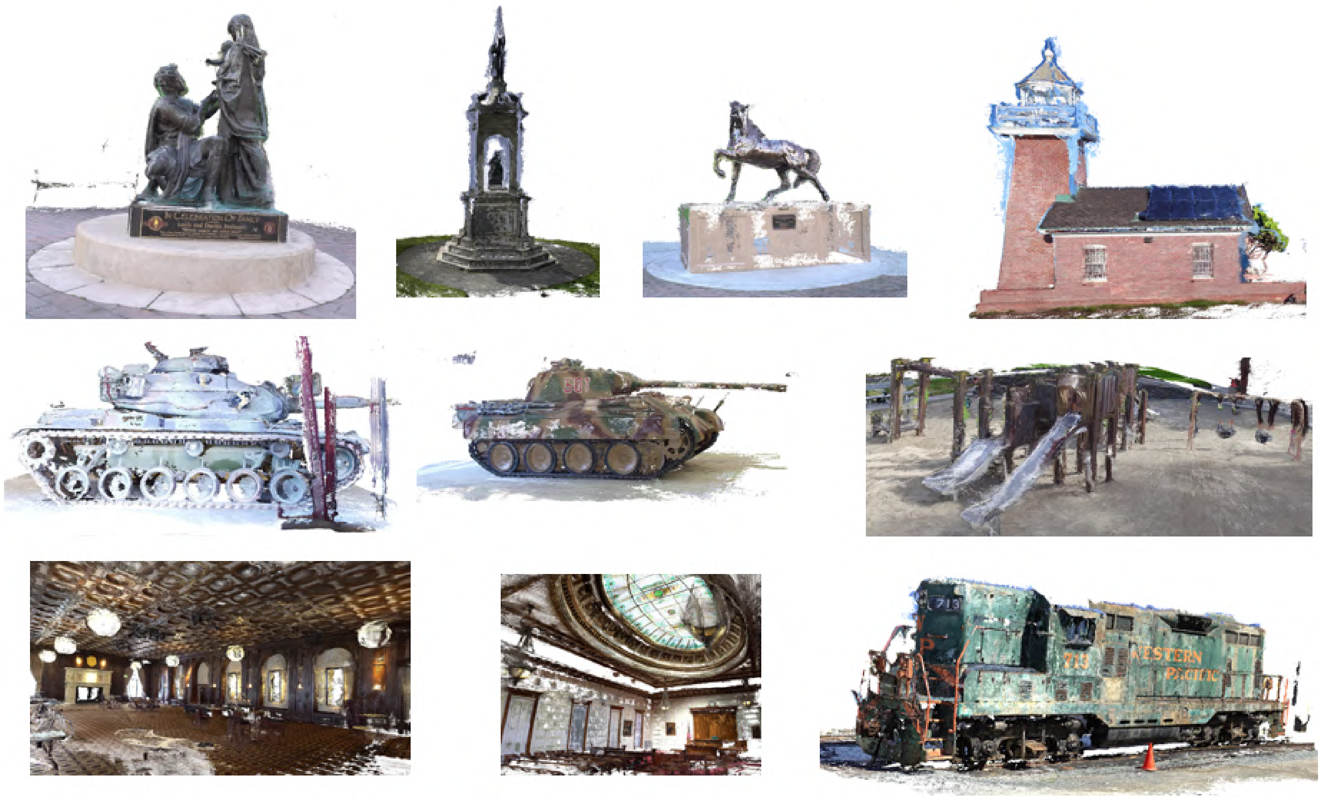
Figure 14: Reconstruction results on Tanks & Temples dataset [24].

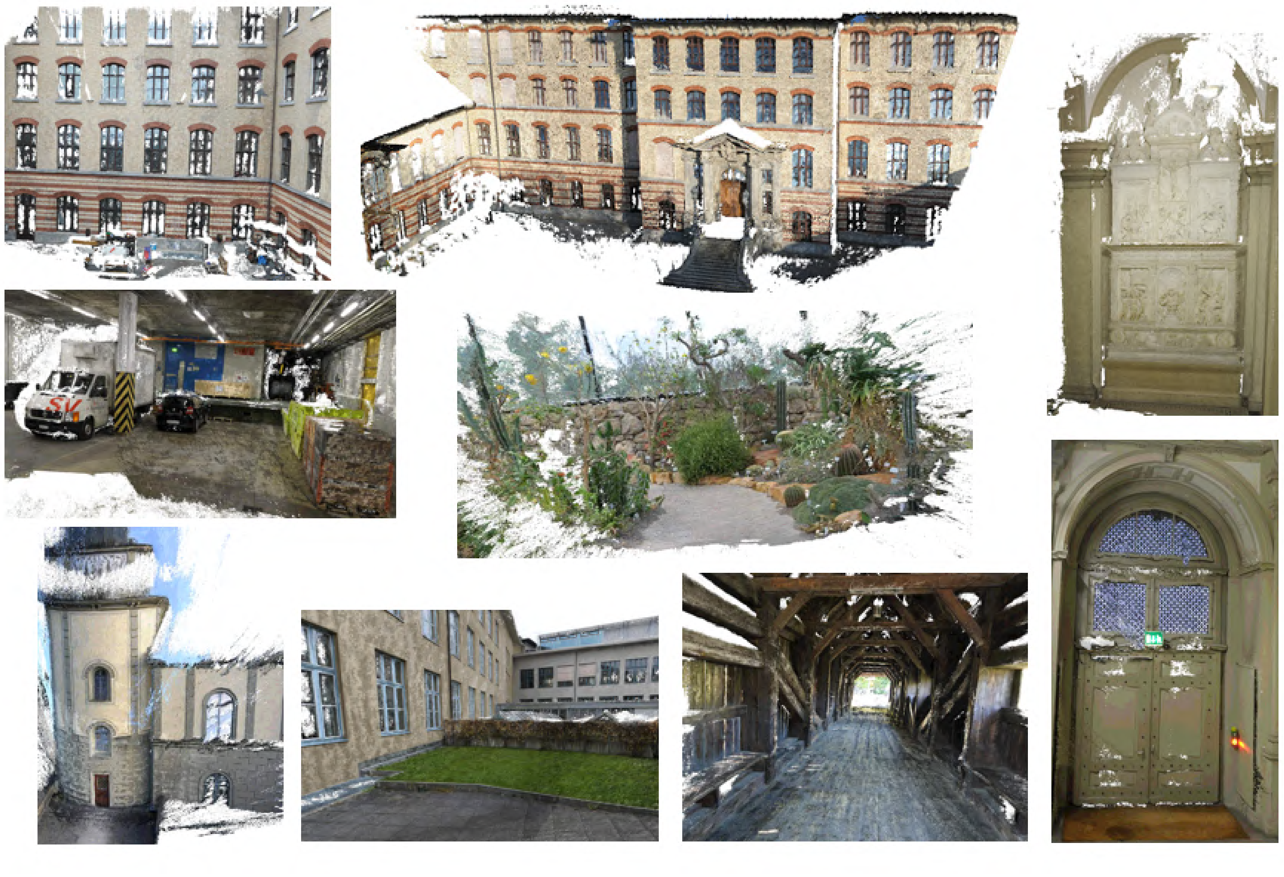

Figure 15: Reconstruction results on ETH3D benchmark [32].

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 (IJCV)*, 2016. 1, 2, 5, 6, 7, 8, 9, 10, 11, 12

[2] Connelly Barnes, Eli Shechtman, Adam Finkelstein, and Dan B Goldman. PatchMatch: A randomized correspondence algorithm for structural image editing. *ACM Transactions on Graphics (SIGGRAPH)*, 2009. 1, 3

[3] Michael Bleyer, Christoph Rhemann, and Carsten Rother. Patchmatch stereo - stereo matching with slanted support windows. In *British Machine Vision Conference (BMVC)*, 2011. 3, 5, 9

[4] Neill D. F. Campbell, George Vogiatzis, Carlos Hernández, and Roberto Cipolla. Using multiple hypotheses to improve depth-maps for multi-view stereo. In *European Conference on Computer Vision (ECCV)*, 2008. 6

[5] Jia-Ren Chang and Yong-Sheng Chen. Pyramid stereo matching network. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018. 2

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

[7] Shuo Cheng, Zexiang Xu, Shilin Zhu, Zhuwen Li, Li Erran Li, Ravi Ramamoorthi, and Hao Su. Deep stereo using adaptive thin volume representation with uncertainty awareness. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020. 1, 2, 4, 5, 6, 7, 9

[8] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using RNN encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014. 2

[9] Robert Collins. A space-sweep approach to true multi-image matching. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 1996. 2, 4

[10] Jifeng Dai, Haozhi Qi, Yuwen Xiong, Yi Li, Guodong Zhang, Han Hu, and Yichen Wei. Deformable convolutional networks. In *International Conference on Computer Vision (ICCV)*, 2017. 2, 3

[11] Shivam Duggal, Shenlong Wang, Wei-Chiu Ma, Rui Hu, and Raquel Urtasun. Deeppruner: Learning efficient stereo matching via differentiable patchmatch. In *International Conference on Computer Vision (ICCV)*, 2019. 2, 3

[12] Zehua Fu and Mohsen Ardabilian Fard. Learning confidence measures by multi-modal convolutional neural networks. In *Winter Conf. on Applications of Computer Vision (WACV)*, 2018. 9

[13] Simon Fuhrmann, Fabian Langguth, and Michael Goesele. Mve-a multi-view reconstruction environment. In *GCH*, 2014. 7

[14] Yasutaka Furukawa and Jean Ponce. Carved visual hulls for image-based modeling. In *European Conference on Computer Vision (ECCV)*, 2006. 2

[15] Yasutaka Furukawa and Jean Ponce. Accurate, dense, and robust multiview stereopsis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2010. 2, 6, 7

[16] Silvano Galliani, Katrin Lasinger, and Konrad Schindler. Massively parallel multiview stereopsis by surface normal diffusion. In *International Conference on Computer Vision (ICCV)*, 2015. 1, 2, 3, 6, 7, 8, 9

[17] Xiaodong Gu, Zhiwen Fan, Siyu Zhu, Zuozhuo Dai, Feitong Tan, and Ping Tan. Cascade cost volume for high-resolution multi-view stereo and stereo matching. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020. 1, 2, 4, 5, 6, 7, 9

[18] Xiaoyang Guo, Kai Yang, Wukui Yang, Xiaogang Wang, and Hongsheng Li. Group-wise correlation stereo network. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. 10

[19] Tak-Wai Hui, Chen Change Loy, and Xiaoou Tang. Depth map super-resolution by deep multi-scale guidance. In *European Conference on Computer Vision (ECCV)*, 2016. 2, 5

[20] Mengqi Ji, Juergen Gall, Haitian Zheng, Yebin Liu, and Lu Fang. Surfacenet: An end-to-end 3D neural network for multiview stereopsis. In *International Conference on Computer Vision (ICCV)*, 2017. 2, 5, 6

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

[22] 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 *International Conference on Computer Vision (ICCV)*, 2017. 2, 10

[23] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *CoRR*, 2015. 6

[24] Arno Knapitsch, Jaesik Park, Qian-Yi Zhou, and Vladlen Koltun. Tanks and temples: Benchmarking large-scale scene reconstruction. *ACM Transactions on Graphics*, 2017. 1, 2, 5, 7, 8, 11, 13

[25] Zhaoxin Li, Kuanquan Wang, Wangmeng Zuo, Deyu Meng, and Lei Zhang. Detail-preserving and content-aware variational multi-view stereo reconstruction. *Transactions on Image Processing (TIP)*, 2016. 2

[26] Tsung-Yi Lin, Piotr Dollár, Ross B. Girshick, Kaiming He, Bharath Hariharan, and Serge J. Belongie. Feature pyramid networks for object detection. *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. 3

[27] Alex Locher, Michal Perdoch, and Luc Van Gool. Progressive prioritized multi-view stereo. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 2

[28] Keyang Luo, Tao Guan, Lili Ju, Haipeng Huang, and Yawei Luo. P-MVSNet: Learning patch-wise matching confidence aggregation for multi-view stereo. In *International Conference on Computer Vision (ICCV)*, 2019. 1, 2, 4, 5, 6, 7, 9

[29] Pierre Moulon, Pascal Monasse, Romuald Perrot, and Renaud Marlet. Openmvg: Open multiple view geometry. In *International Workshop on Reproducible Research in Pattern Recognition*, 2016. 6

[30] Adam Paszke, S. Gross, Francisco Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, Alban Desmaison, Andreas Köpf, E. Yang, Zach DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy,

B. Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. *ArXiv*, 2019. 6

[31] Johannes Lutz Schönberger and Jan-Michael Frahm. Structure-from-Motion Revisited. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 1, 2, 4, 7

[32] Thomas Schöps, Johannes L. Schönberger, Silvano Galliani, Torsten Sattler, Konrad Schindler, Marc Pollefeys, and Andreas Geiger. A multi-view stereo benchmark with high-resolution images and multi-camera videos. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. 1, 2, 5, 7, 8, 11, 14

[33] Akihito Seki and Marc Pollefeys. Patch based confidence prediction for dense disparity map. In *British Machine Vision Conference (BMVC)*, 2016. 9

[34] Sudipta N. Sinha, Philippos Mordohai, and Marc Pollefeys. Multi-view stereo via graph cuts on the dual of an adaptive tetrahedral mesh. In *International Conference on Computer Vision (ICCV)*, 2007. 2

[35] Engin Tola, Christoph Strecha, and Pascal Fua. Efficient large-scale multi-view stereo for ultra high-resolution image sets. *Machine Vision and Applications*, 2012. 6

[36] Ali Osman Ulusoy, Michael J. Black, and Andreas Geiger. Semantic multi-view stereo: Jointly estimating objects and voxels. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. 2

[37] Haofei Xu and Juyong Zhang. AANet: Adaptive aggregation network for efficient stereo matching. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020. 2, 5

[38] Qingshan Xu and Wenbing Tao. Multi-scale geometric consistency guided multi-view stereo. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. 1, 2, 3, 4, 9

[39] Qingshan Xu and Wenbing Tao. Learning inverse depth regression for multi-view stereo with correlation cost volume. In *AAAI*, 2020. 1, 2, 3, 4, 5, 6, 7, 9, 10

[40] Qingshan Xu and Wenbing Tao. PVSNet: Pixelwise visibility-aware multi-view stereo network. *ArXiv*, 2020. 2, 4, 5, 7, 9

[41] Jiayu Yang, Wei Mao, Jose M. Alvarez, and Miaomiao Liu. Cost volume pyramid based depth inference for multi-view stereo. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020. 1, 2, 5, 6, 7, 9

[42] Yao Yao, Zixin Luo, Shiwei Li, Tian Fang, and Long Quan. MVSNet: Depth inference for unstructured multi-view stereo. In *European Conference on Computer Vision (ECCV)*, 2018. 1, 2, 4, 5, 6, 7, 9, 10

[43] Yao Yao, Zixin Luo, Shiwei Li, Tianwei Shen, Tian Fang, and Long Quan. Recurrent MVSNet for high-resolution multi-view stereo depth inference. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. 1, 2, 3, 4, 6, 7

[44] Zehao Yu and Shenghua Gao. Fast-MVSNet: Sparse-to-dense multi-view stereo with learned propagation and gauss-newton refinement. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020. 1, 2, 6, 7

[45] Enliang Zheng, Enrique Dunn, Vladimir Jojic, and Jan-Michael Frahm. Patchmatch based joint view selection and depthmap estimation. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014. 1