

PatchMatch-RL: Deep MVS with Pixelwise Depth, Normal, and Visibility

Jae Yong Lee¹Joseph DeGol²Chuhang Zou^{*3}Derek Hoiem¹¹University of Illinois at Urbana-Champaign

{lee896, dhoiem}@illinois.edu

²Microsoft

joseph.degol@microsoft.com

³Amazon Go

zouchuha@amazon.com

Abstract

Recent learning-based multi-view stereo (MVS) methods show excellent performance with dense cameras and small depth ranges. However, non-learning based approaches still outperform for scenes with large depth ranges and sparser wide-baseline views, in part due to their PatchMatch optimization over pixelwise estimates of depth, normals, and visibility. In this paper, we propose an end-to-end trainable PatchMatch-based MVS approach that combines advantages of trainable costs and regularizations with pixelwise estimates. To overcome the challenge of the non-differentiable PatchMatch optimization that involves iterative sampling and hard decisions, we use reinforcement learning to minimize expected photometric cost and maximize likelihood of ground truth depth and normals. We incorporate normal estimation by using dilated patch kernels and propose a recurrent cost regularization that applies beyond frontal plane-sweep algorithms to our pixelwise depth/normal estimates. We evaluate our method on widely used MVS benchmarks, ETH3D and Tanks and Temples (TnT). On ETH3D, our method outperforms other recent learning-based approaches and performs comparably on advanced TnT.

1. Introduction

Multi-view stereo (MVS) aims to reconstruct 3D scene geometry from a set of RGB images with known camera poses, with many important applications such as robotics [25], self-driving cars [8], infrastructure inspection [7, 13], and mapping [31]. Non-learning based MVS methods [5, 26, 32, 34, 41] evolved to support pixelwise estimates of depths, normals, and source view selection, with PatchMatch based iterative optimization and cross-image consistency checks. Recent learning-based MVS methods [12, 15, 16, 39, 40] tend to use frontal plane sweeps, evaluating the same set of depth candidates for each pixel based on the same images. The trainable photometric scores and cost-volume regularization of the learning-based methods leads to excel-

*Work done outside of Amazon

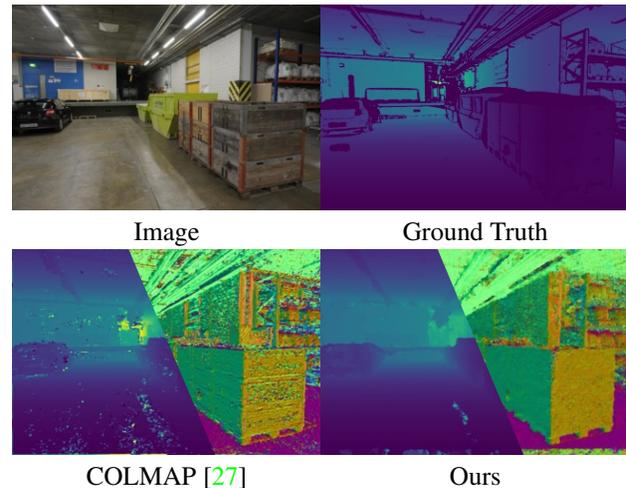
Code available at <https://github.com/leejaeyong7/patchmatch-rl>

Figure 1. We propose **PatchMatch-RL**, an end-to-end trainable PatchMatch-based MVS approach that combines advantages of trainable costs and regularizations with pixelwise estimates of depth, normal, and visibility. The left half of the bottom images is the depth, and the right half is the normals. We show that our method can achieve smoother and more complete depth and normal map estimation over the existing approach (COLMAP).

lent performance with dense cameras and small depth ranges, as evidenced in the DTU [2] and Tanks-and-Temples (TnT) benchmarks [18], but the pixelwise non-learning based approach outperforms for scenes with large depth ranges and slanted surfaces observed with sparser wide-baseline views, as evidenced in the ETH3D benchmark [28].

Our paper aims to incorporate pixelwise depth, normal, and view estimates into an end-to-end trainable system with advantages from both approaches:

- **Pixelwise depth and normal** prediction efficiently models scenes with large depth ranges and slanted surfaces.
- **Pixelwise view selection** improves robustness to occlusion and enables reconstruction from sparser images.
- **Learned photometric cost** functions improve correspondence robustness.

- **Learned regularization and contextual inference** enable completion of textureless and glossy surfaces.

One challenge is that PatchMatch optimization and pixelwise view selection involve iterative sampling and hard decisions that are not differentiable. We propose a reinforcement learning approach to minimize expected photometric cost and maximize discounted rewards for reaching a good final solution. Our techniques can also be used to enable learning for other PatchMatch applications (e.g. [3, 14, 21]), though we focus on MVS only. Estimating 3D normals of pixels is also challenging because convolutional features tend to be smooth so that neighboring cells add little new information, and patch-wise photometric costs are memory intensive. We find that with shallower feature channels and dilated patch kernels, we effectively estimate pixel normals. A third challenge is how to perform regularization or global inference. Each pixel has its own depth/normal estimate, so cost-volume based regularization does not apply. We propose a recurrent cost regularization that updates a hidden state via message passing that accounts for depth/normal similarities between pixels.

In summary, our main **contribution** is an end-to-end trainable PatchMatch-based MVS approach that combines advantages of trainable costs and regularizations with pixelwise estimates, requiring multiple innovations:

- Reinforcement learning approach to train end-to-end within a PatchMatch sampling based optimization.
- Use of normal estimates in learning-based MVS, enabled by trainable PatchMatch optimization and CNN patch features.
- Depth/normal regularization that applies beyond frontal plane-sweep algorithms; e.g. to our pixelwise depth/normal estimates.

In experiments, our system outperforms other recent learning-based methods on ETH3D and performs similarly on TnT, and our ablation study validates the importance of pixelwise normal and view selection estimates.

2. Related Works

Given correct scene geometry, the pixels that correspond to a surface patch in different calibrated cameras can be determined, and their appearance patterns will be similar (“photometrically consistent”). This core idea of multi-view stereo (MVS) leads to an array of formulations, optimization algorithms, and refinements. We focus on our work’s direct lineage, referring the interested reader to a survey/tutorial [9] and paper list [1] for more complete background and coverage.

The first and simplest formulation is to assign each pixel to one of a set of candidate disparities or depth values [22]. The locally best assignment can be determined by filtering across rows in rectified images, and surface smoothness priors can be easily incorporated within this ordered labeling problem. However, per-view depth labeling has many shortcomings in a wide-baseline MVS setting: (1) depth maps do

not align in different views, making consistency checking and fusion more difficult; (2) depth for oblique surfaces is not constant, degrading matching of intensity patches; and (3) the range of depth values may be large, so that large steps in depth are needed to feasibly evaluate the full range. Further, occlusion and partially overlapping images demand more care in evaluating photometric consistency.

These difficulties led to a reformulation of MVS as solving for a depth, normal, and view selection for each pixel in a reference image [27, 41]. The view selection identifies which other source images will be used to evaluate photometric consistency. This more complex formulation creates a challenging optimization problem, since each pixel has a 4D continuous value (depth/normal) and binary label vector (view selection). PatchMatch [3, 5, 27] is well-suited for the depth/normal optimization, since it employs a hypothesize-test-propagate framework that is ideal for efficient inference when labels have a large range but are approximately piecewise constant in local neighborhoods. The pixelwise PatchMatch formulations have been refined with better propagation schemes [32], multi-scale features [32], and plane priors [26, 34]. Though this line of work addresses the shortcomings of the depth labeling approach, it often fails to reconstruct smooth or glossy surfaces where photometric consistency is uninformative, mainly due to the challenge of incorporating global priors, which is addressed in part by Kuhn et al.’s post-process trainable regularization [19]. Also, though handcrafted photometric consistency functions, such as bilaterally weighted NCC, perform well in general, learned functions can potentially outperform by being context-sensitive.

Naturally, the first inroads to fully trainable MVS also followed the simplistic depth labeling formulation [15, 16, 36], which comfortably fits the CNN forte of learning features, performing inference over “cost volumes” (features or scores for each position/label), and producing label maps. But despite improvements such as using recurrent networks [37] to refine estimates, coarse-to-fine reconstruction [39], visibility maps [35], and attention-based regularization [23], many of the original drawbacks of the depth labeling formulation persist.

Thus, we now have two parallel branches of MVS state-of-the-art: (1) complex hand-engineered formulations with PatchMatch optimization that outperform for large-scale scene reconstruction from sparse wide-baseline views; and (2) deep network depth-labeling formulations that outperform for smaller scenes, smooth surfaces, and denser views. Differentiation-based learning and sampling-based optimization are not easily reconciled with refinements or combinations of existing approaches. Duggal et al. [8] propose a differentiable PatchMatch that optimizes softmax-weighted samples, instead of argmax, and use it to prune the depth search space to initialize depth labeling. We use their idea of one-hot filter banks to perform propagation but use an expectation based loss that sharpens towards argmax during training to enable argmax inference. The very recent Patch-

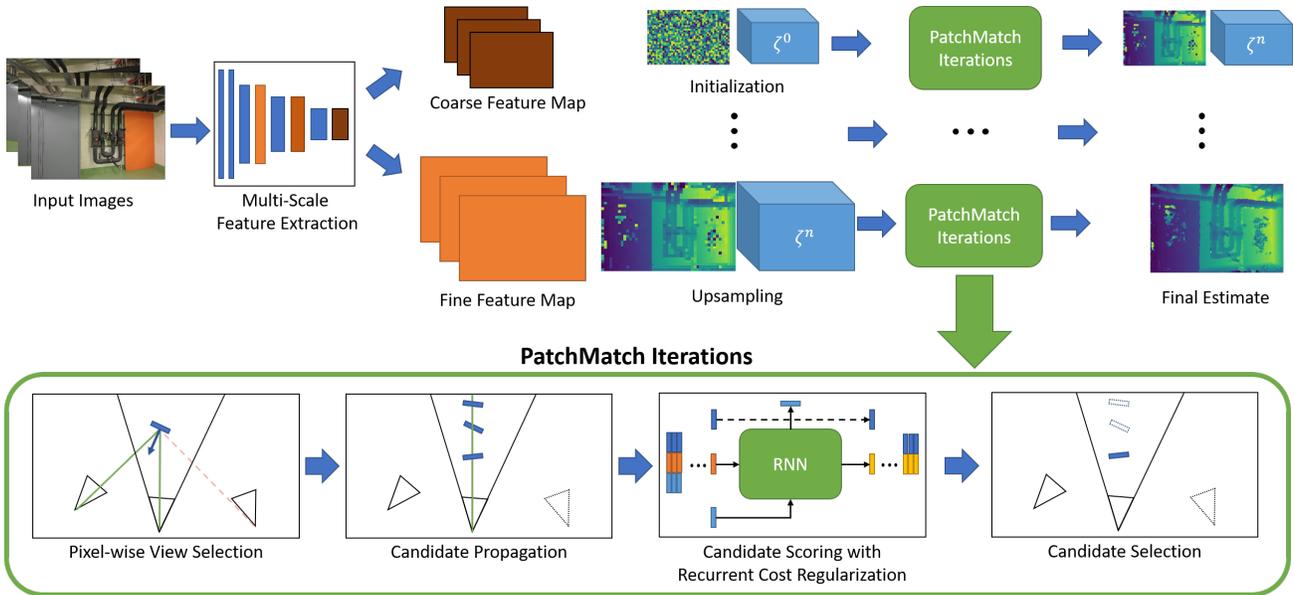


Figure 2. **Architecture overview:** We first extract multi-scale features using CNNs with shared weights. We then perform coarse-to-fine estimation, with correlation of feature maps at corresponding scales used to evaluate photometric costs and perform view selection. At the coarsest stage, we initialize pixelwise oriented points (depths/normals) and associated hidden states per plane. Then, a series of PatchMatch iterations updates the points and hidden state maps. The PatchMatch iteration consists of four stages: (1) pixelwise view selection; (2) candidate propagation; (3) candidate scoring with recurrent cost regularization; and (4) candidate selection. The current solution is then upsampled as an input to the finer level, and this continues until oriented point estimates at the finest level are fused from all images.

matchNet [30] minimizes a sum of per-iteration losses and employs a one-time prediction of visibility (soft view selection). We use reinforcement learning to train view selection and minimize the loss of the final depth/normal estimates. Our work is the first, to our knowledge, to propose an end-to-end trainable formulation that combines the advantages of pixelwise depth/normal/view estimates and PatchMatch optimization with deep network learned photometric consistency and refinement.

3. PatchMatch-RL MVS

We propose PatchMatch-RL, an end-to-end learning framework that uses PatchMatch for Multi-View Stereo (MVS) reconstruction. Figure 2 shows an overview of our approach. Given a set of images \mathcal{I} and its corresponding camera poses $\mathcal{C} = (\mathcal{K}, \mathcal{E})$ with intrinsic \mathcal{K} and extrinsic $\mathcal{E} = [\mathcal{R}, t]$ matrices, our goal is to recover the depths (and normals) of the reference image $I_{ref} \in \mathcal{I}$ using a set of selected source images $I_{src} \subset \mathcal{I}$ that overlap with I_{ref} .

Rather than solving only for depth, we also estimate surface normals, which enables propagating hypotheses and comparing spatially distributed features between reference and source images along the local plane. Surface normal estimation improves depth estimates for oblique surfaces and is also useful for consistency checks, surface modeling, and other downstream processing.

Our estimation proceeds coarse-to-fine. At the coarsest level, estimates are randomly initialized and then refined through a series of PatchMatch iterations that consist of

pixelwise view selection, candidate propagation, regularized cost computation, and candidate update. Resulting estimates are then upsampled and further refined, this continues until the finest layer, after which all depth estimates are fused into a 3D point cloud.

3.1. Initialization

For each level in the coarse-to-fine optimization, we extract CNN features for the reference and source images using a Feature Pyramid Network (FPN) [20]. For memory efficiency, the number of output channels varies per scale, with shallower feature channels in the higher-resolution feature maps. \mathcal{F}_p^s denotes the feature vector for pixel p at image s .

Our goal is to solve for an oriented point ω_p , consisting of a plane-camera distance δ_p and normal \mathbf{n}_p , for each pixel p in I_{ref} . Pixel depth d_p is related to δ_p through $d_p = -\delta_p / (\mathbf{n}_p \cdot \mathcal{K}^{-1} \cdot p)$. The depth d_p is sampled uniformly from the inverse depth range as: $d_p^0 \sim 1/\mathcal{U}(\frac{1}{d_{max}}, \frac{1}{d_{min}})$, with d_{min} and d_{max} specifying the depth range. Sampling from the inverted range prioritizes depths closer to the camera center, as shown effective by Gallup *et al.* [10]. The per-pixel normal \mathbf{n}_p is initialized independently of depth by sampling from a 3D Gaussian and applying L2 normalization [24]. The normal vector is reversed if it faces the same direction as the pixel ray.

3.2. Feature Correlation

The feature maps can be differentially warped [36] according to the pixelwise plane homographies from refer-



Figure 3. **Estimated Visibilities.** The leftmost image corresponds to the reference image, and the right two images are the source images. We color-code the corresponding region to be estimated as visible in the last PatchMatch iteration. The estimated visibility matches precisely with the actual visibility. (Best viewed in color.)

ence image r to source image s as $H_{\omega_p}^{r \rightarrow s} = \mathcal{K}_s \cdot (\mathcal{R}_{r \rightarrow s} - \frac{t_{r \rightarrow s} \mathbf{n}_p^T}{\delta_p}) \cdot \mathcal{K}_r^{-1}$. With support window $\mathcal{W}_p^{\alpha, \beta}$ of size α and dilation β centered at p , we define the correlation value $\mathcal{G}_{\omega_p}^s$ of the oriented point ω_p as the attention-aggregated group-wise correlation for matching feature vectors in the source image:

$$\mathcal{A}_p^q = \sigma\left(\frac{\mathcal{F}_p^r \cdot h}{\sqrt{\|\mathcal{F}_p^r\|_2}}\right)_q, q \in \mathcal{W}_p^{\alpha, \beta}$$

$$\mathcal{G}_{\omega_p}^s = \sum_q \mathcal{A}_p^q \cdot (\mathcal{F}_q^r \otimes \mathcal{F}_{H_{\omega_p}^{r \rightarrow s}, q}^s).$$

We denote group-wise feature vector correlation [33] as \otimes , scaled dot-product attention for supporting pixel q on center pixel p by the reference feature map as \mathcal{A}_p^q , and the attentional feature projection vector as h , implemented as a 1x1 convolution. The resulting $\mathcal{G}_{\omega_p}^s$ represents the similarity of the features centered at p in the reference image and the corresponding features in the source image, according to ω_p .

In preliminary experiments, our estimation of normals \mathbf{n}_p was poor and did not improve depth estimation. The problem was that the smoothness of features prevented a 3x3 patch from providing much additional information. Making larger patches was not practical due to memory constraints. This problem was solved through use of dilation ($\beta = 3$), and we further reduced memory usage by producing shallower feature channels.

3.3. Pixel-wise View Selection

Based on Schönberger et al. [27], we compute scale, incident-angle, and triangulation angle difference based geometric priors for each source image s for each ω_p . Instead of hand-crafting the prior function, we concatenate the priors with the feature correlations $\mathcal{G}_{\omega_p}^s$ and use a multi-layered perceptron (MLP) to predict a pixel-wise visibility estimate, denoted $\hat{\mathcal{V}}_p^s \in [0, 1]$. Figure 3 shows an example of the estimated visibilities in the source images.

We then sample N -views based on the L1 normalized probability distribution over $\hat{\mathcal{V}}_p^s$ for each pixel, to obtain a sampled set of views, \mathcal{V}_p . The visibility probabilities are further used to compute a weighted sum of feature correlations across views.

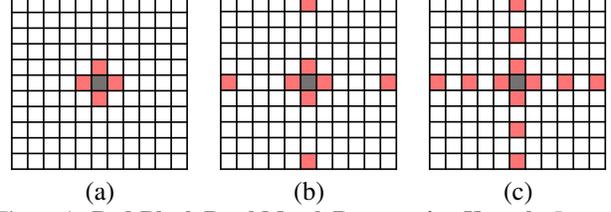


Figure 4. **Red-Black PatchMatch Propagation Kernels.** In each kernel, the grey square indicates the pixel to be updated. The red squares indicate the neighboring pixels that provide PatchMatch a set of candidate oriented points for the gray pixel. We use kernel (c) for the coarsest level and kernel (b) for the finer levels.

3.4. Candidate Propagation

The oriented point map ω^t at the t -th PatchMatch iteration is propagated according to the propagation kernel. A common kernel is the Red-Black propagation kernel by Galliani et al. [10], as illustrated in Figure 4. We let $\psi_{:,p}^t = \{\omega_q^t \mid q \in K(p)\} \cup \{\omega_p^{t,prt}\}$ denote the set of candidate oriented points obtained by propagation kernel K at pixel p and by random perturbation of the current candidate. The propagation can be applied using a series of convolutional filters of one-hot encodings, with one values in positions that correspond to each neighbor, as defined by K . The visibility-weighted feature correlations for each candidate are computed as $\mathcal{G}_{\psi_{k,p}}^{\mathcal{V}} = \frac{\sum_{v \in \mathcal{V}_p} \hat{\mathcal{V}}_p^v \mathcal{G}_{\psi_{k,p}}^v}{\sum_{v \in \mathcal{V}_p} \hat{\mathcal{V}}_p^v}$.

3.5. Candidate Regularized Cost and Update

Existing learning-based cost regularization methods, such as 3D convolution on spatially aligned cost volume [36] or k -nearest neighbor based graph convolutions [6], exploit ordinal relationships between neighboring label maps. However, there is no consistent relationship between candidates for ω_p or for candidates of neighboring pixels. Instead, we get insight from loopy Belief-Propagation (LBP), where each node’s belief is iteratively updated by message-passing from the neighboring nodes, so that confidently labeled nodes propagate to less confident neighbors. We represent beliefs for each candidate as hidden states $\zeta_{\psi_{k,p}}^t$, and use a recurrent neural network (RNN) to estimate regularized score $\mathcal{Z}_{\psi_{k,p}}$ and updated hidden state $\zeta_{\psi_{k,p}}^{t+1}$. Figure 5 illustrates the process.

Similar to LBP, we compute pairwise neighborhood smoothness [4] of the candidate with respect to the current label, $\{M(\psi_{k,p}, \omega_q) \mid q \in \mathcal{N}(p)\}$, where $M(\omega_p, \omega_q) = \text{dist}(\omega_p, q) + \text{dist}(\omega_q, p)$ is the sum of distances between each oriented point and the plane parameterized by the other oriented point. We append the smoothness terms to the weighted feature correlation $\mathcal{G}_{\psi_{k,p}}^{\mathcal{V}}$ as an input to the RNN. The RNN can then aggregate the confidences (represented by feature correlations) over similar oriented points.

The per-pixel candidates and corresponding hidden states

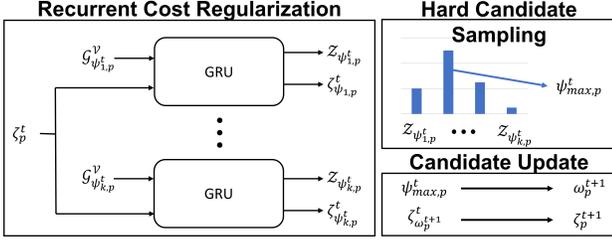


Figure 5. **Recurrent Cost Regularization.** Given the hidden state ζ_p^t of each pixel p in the t -th iteration and the visibility-weighted feature correlations of each propagated candidate $G_{\psi_{i,p}^t}^v$, we use a Gated Recurrent Unit (GRU) module to estimate the regularized cost $Z_{\psi_{i,p}^t}$ and updated hidden state $\zeta_{\psi_{i,p}^t}^t$ for each plane candidate. Then, the best candidate $\psi_{max,p}^t$ for the next iteration is hard-sampled according to the regularized costs, replacing the current oriented point ω_p^t at p , and the corresponding hidden states of the pixel ζ_p^t are updated using the corresponding sampled candidate $\zeta_{\omega_p^{t+1}}^t$.

are updated by:

$$\begin{aligned}\omega_p^{t+1} &= \psi_{k,p}^t \sim Z_{\psi_{i,p}^t}^t \\ \zeta_p^{t+1} &= \zeta_{\omega_p^{t+1}}^t.\end{aligned}$$

In inference, the sampling of ω_p is $\arg \max$; in training, the sampling hardens from probabilistic to $\arg \max$ as training progresses. The updated hidden states are used as an input to the recurrent cost regularization module in the next PatchMatch iteration.

3.6. Coarse-to-Fine PatchMatch and Fusion

The estimated map of oriented points ω^t and the corresponding hidden states ζ^t are upsampled as an input to the finer level PatchMatch iteration using nearest neighbor interpolation. The ω of the finest level are fused together into a 3D point cloud by following the method used by other MVS systems [10, 27, 36]. First, consistency is checked for each reference image with the source views using reprojection distance, relative depth distance, and normal consistency. Then, we reproject the mean value of N -view consistent depths into the world space to obtain consensus points.

4. PatchMatch-RL Training

It is challenging to make PatchMatch MVS end-to-end trainable. The $\arg \max$ based hard decisions/sampling required for PatchMatch update and view selection is non-differentiable, and the incorporation of normal estimates with soft- $\arg \max$ causes depth and normal to depend on each other. We propose a reinforcement learning approach to jointly learn the candidate cost and visibility estimation parameters.

We use $\mathcal{V}^{\theta_V}(\omega_p)$ to denote the pixel-wise visibility estimation function, parameterized by θ_V , that outputs visibility score \hat{V}_p^s for each source image s given images \mathcal{I} and cameras \mathcal{C} . We use $\mathcal{S}^{\theta_S}(\psi_p)$ to denote a matching score function,

parameterized by θ_S , that produces plane candidate score Z_{ψ_p} for each ψ_p given \mathcal{I}, \mathcal{C} and selected views \mathcal{V}_p . Our formulation contains two agents: one selects views and the other selects the candidates.

4.1. Reward Function

We define the reward $r^t = \mathcal{N}(\omega^t; \omega^*, \sigma_\omega)$ as a probability of observing the oriented point ω^t from distribution given ground truth oriented point value ω^* in iteration t . We define the distribution as a joint independent normal distribution of depth and normal of pixel p :

$$\mathcal{N}(\omega_p^t; \omega_p^*, \sigma_\omega) = \mathcal{N}(n_p^t; n_p^*, \sigma_n) \cdot \mathcal{N}(d_p^t; d_p^*, \sigma_d). \quad (1)$$

We let the expected future reward be a γ -discounted sum of future rewards: $G^t = \sum_{t' \geq t} \gamma^{t'-t} r^{t'}$. We formulate the gradient of the reward as a negation of the gradient of cross-entropy between the step-wise reward $\mathcal{N}(\omega^t; \omega^*, \sigma_\omega)$ and an agent $\pi_\theta(a^t, s^t)$, according to the REINFORCE algorithm as:

$$\begin{aligned}\nabla_\theta J &= \mathbb{E}_{\pi_\theta} [Q^{\pi_\theta}(s, a) \nabla_\theta \log \pi_\theta(a | s)] \\ &= \sum_t \nabla_\theta \ln \pi_\theta^t(a^t | s^t) G^t \\ &= \sum_t \sum_{t' \geq t} \nabla_\theta \gamma^{t'-t} (\mathcal{N}(\omega^{t'}; \omega^*, \sigma_\omega) \log \pi_\theta). \quad (2)\end{aligned}$$

The sampling can be done in two ways: the categorical distribution, which makes the policy approximate the expectation of the distribution; or $\arg \max$, which makes the policy the greedy solution. As an exploration versus exploitation strategy, we employ a decaying ϵ -greedy approach where we sample candidates using (1) expectation by probability of ϵ or (2) using $\arg \max$ by probability of $1 - \epsilon$. We also apply a decaying reward of $\tau \cdot \mathcal{N}(d_p^t; d_p^*, \sigma_d)$.

Below, we describe the policy of each agent. We use $\mathbb{S}_V, \mathbb{A}_V, \pi_V, \mathbb{R}_V$, and $\mathbb{S}_S, \mathbb{A}_S, \pi_S, \mathbb{R}_S$ to denote the state, action, policy and reward space of the view selection and candidate selection agents respectively. For simplicity, we use $s^t \in \mathbb{S}$, $a^t \in \mathbb{A}$, and $r^t \in \mathbb{R}$ to denote the corresponding agent's state, action, and reward in the t -th iteration that apply to a particular pixel.

4.2. Learning Photometric Cost

For the candidate selecting agent, the state space is the set of candidate plane parameters ψ : for each oriented point ω_p , and the the action space is the selection of a candidate label for each pixel in each iteration according to the parameterized photometric cost function $\mathcal{S}^{\theta_S}(\omega_p)$. The probability of selecting each candidate is defined as a softmax distribution based on the photometric cost of each plane candidate, and the stochastic policy π_S samples from this distribution:

$$\pi_S(a^t | s^t) = \omega^t \sim \frac{e^{-\mathcal{S}^{\theta_S}(\psi_q^t)}}{\sum_{q \in K} e^{-\mathcal{S}^{\theta_S}(\psi_q^t)}} \quad (3)$$



Ref. Image GT. Depth COLMAP Ours

Figure 6. **Qualitative comparison against COLMAP on the ETH3D high-resolution benchmark.** From the left, reference image, ground truth depth, depth estimate from COLMAP, normal estimate from COLMAP, depth estimate of our model, and normal estimate of our model. All of the depth maps share the same color scale based on the ground truth depth ranges. We show that our estimated depths and normals are more complete than COLMAP.

The parameters can be learned via gradient ascent through the negative cross-entropy between the probability distribution of the candidates given ground truth and the probability distribution of the candidates estimated by photometric cost function:

$$\nabla_{\theta_S} \mathcal{N}_{\omega^t} \log \pi_S = \nabla_{\theta_S} \sum_{k \in K} \mathcal{N}_{\psi_k^t} \cdot \log \left(\frac{e^{-S^{\theta_S}(\psi_k^t)}}{\sum_{j \in K} e^{-S^{\theta_S}(\psi_j^t)}} \right)$$

where $\mathcal{N}_{\psi_k^t} = \mathcal{N}(\psi_k^t; \omega^*, \sigma_\psi)$ represents the probability of observing the candidate ψ_k^t according to the ground truth.

4.3. Learning View Selection

For the view selection agent, the state space contains the set of source images; the action space is a selection of N images among the source images for each iteration; and the policy uses the parameterized view selection function $\mathcal{V}(\omega_p^t)$ to estimate the visibility ($\forall s \in I_{src}$), $\hat{\mathcal{V}}^s$. The stochastic policy $\pi_{\mathcal{V}}$ is:

$$\pi_{\mathcal{V}}(a^t | s^t) = v \sim \frac{\hat{\mathcal{V}}^v}{\sum_{s \in I_{src}} \hat{\mathcal{V}}^s} \quad (4)$$

and the gradient:

$$\begin{aligned} \nabla_{\theta_{\mathcal{V}}} \log \pi_{\mathcal{V}} &= \nabla_{\theta_{\mathcal{V}}} \log \left(\frac{\sum_{v \in N} \hat{\mathcal{V}}^v}{\sum_{s \in I_{src}} \hat{\mathcal{V}}^s} \right) \\ &\approx \nabla_{\theta_{\mathcal{V}}} \left(\log \left(\sum_{v \in N} \hat{\mathcal{V}}^v \right) - \log \left(\sum_{m \in (N \cup M)} \hat{\mathcal{V}}^m \right) \right). \end{aligned}$$

For robustness of training, we include only the selected N views and worse M views in the denominator to prevent minimizing the probabilities of good but unselected views. This incentivizes training to assign more visibility to good views than bad views (that do not view the point corresponding to the reference pixel).

5. Experiments

We evaluate our work on two large-scale benchmarks: Tanks and Temples Benchmark [18] and ETH3D High-Res Multi-View Benchmark [28].

5.1. Training Details

For all experiments, we train using the BlendedMVS dataset [38], which contains a combination of 113 object, indoor, and outdoor scenes with large viewpoint variations. We use the low-res version of the dataset which has a spatial resolution of 768×576 . Throughout training and evaluation, we use $\alpha = 3$ and $\beta = 3$, 3 layers of hidden states \mathcal{H} , $\gamma = 0, 1$ for photometric scorer and view selection scorer respectively, and feature map sizes corresponding to $\frac{1}{8}$, $\frac{1}{4}$ and $\frac{1}{2}$ of the original image size. For training, we use 2, 1, and 1 iterations, and for evaluation we use 8, 2, and 2 iterations for each scale respectively. We use the PatchMatch Kernel K shown in Figure 4(b) for training. As an exploitation versus exploration strategy, we employ a Decaying ϵ -Greedy approach where we either sample candidates proportional to their softmax scores with a probability of ϵ or select the argmax candidate with the probability of $1 - \epsilon$. The initial value of ϵ is 0.9 with an exponential decay of 0.999 per each

Method	Resolution	Time(s)	Test 2cm: Accuracy / Completeness / F1			Test 5cm: Accuracy / Completeness / F1		
			Indoor	Outdoor	Combined	Indoor	Outdoor	Combined
ACMH [32]	3200x2130	546.77	91.1 / 64.8 / 73.9	84.0 / 80.0 / 81.8	89.3 / 68.6 / 75.9	97.4 / 78.0 / 83.7	94.1 / 75.0 / 90.4	96.6 / 87.1 / 85.4
Gipuma [10]	2000x1332	272.81	86.3 / 31.4 / 41.9	78.8 / 45.3 / 55.2	84.4 / 34.9 / 45.2	95.8 / 42.1 / 54.9	93.8 / 54.3 / 67.2	95.3 / 45.1 / 58.0
COLMAP [27]	3200x2130	2245.57	92.0 / 59.7 / 70.4	92.0 / 73.0 / 80.8	92.0 / 63.0 / 73.0	96.6 / 73.0 / 82.0	97.1 / 83.9 / 89.7	96.8 / 75.7 / 84.0
PVSNet [35]	1920x1280	-	65.6 / 78.6 / 70.9	68.8 / 84.3 / 75.7	66.4 / 80.1 / 72.1	82.4 / 87.8 / 84.7	84.5 / 92.7 / 88.2	82.9 / 89.0 / 85.6
PatchmatchNet [30]	2688x1792	491.69	68.8 / 74.6 / 71.3	72.3 / 86.0 / 78.5	69.7 / 77.5 / 73.1	84.6 / 85.1 / 84.7	87.0 / 92.0 / 89.3	85.2 / 86.8 / 85.9
Ours	1920x1280	556.50	73.2 / 70.0 / 70.9	78.3 / 78.3 / 76.8	74.5 / 72.1 / 72.4	88.0 / 83.7 / 85.5	92.6 / 89.0 / 90.5	89.2 / 85.0 / 86.8

Method	Resolution	Time(s)	Train 2cm: Accuracy / Completeness / F1			Train 5cm: Accuracy / Completeness / F1		
			Indoor	Outdoor	Combined	Indoor	Outdoor	Combined
ACMH [32]	3200x2130	486.35	92.6 / 59.2 / 70.0	84.7 / 64.4 / 71.5	88.9 / 61.6 / 70.7	97.7 / 70.1 / 80.5	95.4 / 75.6 / 83.5	96.6 / 72.7 / 81.9
Gipuma [10]	2000x1332	243.34	89.3 / 24.6 / 35.8	83.2 / 25.3 / 37.1	86.5 / 24.9 / 36.4	96.2 / 34.0 / 47.1	95.5 / 36.7 / 51.7	95.9 / 35.2 / 49.2
COLMAP [27]	3200x2130	2102.71	95.0 / 52.9 / 66.8	88.2 / 57.7 / 68.7	91.9 / 55.1 / 67.7	98.0 / 66.6 / 78.5	96.1 / 73.8 / 82.9	97.1 / 69.9 / 80.5
PatchmatchNet [30]	2688x1792	473.92	63.7 / 67.7 / 64.7	66.1 / 62.8 / 63.7	64.8 / 65.4 / 64.2	78.7 / 80.0 / 78.9	86.8 / 73.2 / 78.5	82.4 / 76.9 / 78.7
Ours	1920x1280	555.58	76.6 / 60.7 / 66.7	75.4 / 64.0 / 69.1	76.1 / 62.2 / 67.8	89.6 / 76.5 / 81.4	88.8 / 81.4 / 85.7	90.5 / 78.8 / 83.3

Table 1. **Results on the ETH3D high-resolution MVS benchmark train and test sets.** We do not train on any ETH3D data. Bold denotes the method with the highest F1 score for each setting. Results from several other methods are shown for comparison. We measure the mean time taken for reconstructing each scene (including the fusion stage) using the author provided code on the same hardware. PVSNet results are not available on the train set. Our method outperforms other recent learning-based approaches (PVSNet and PatchmatchNet) in most of the metrics.

Method	Precision / Recall / F1	
	Intermediate	Advanced
CIDER [33]	42.8 / 55.2 / 46.8	26.6 / 21.3 / 23.1
COLMAP [27]	43.2 / 44.5 / 42.1	33.7 / 24.0 / 27.2
R-MVSNet [37]	43.7 / 57.6 / 48.4	31.5 / 22.1 / 24.9
CasMVSNet [11]	47.6 / 74.0 / 56.8	29.7 / 35.2 / 31.1
AttMVS [23]	61.9 / 58.9 / 60.1	40.6 / 27.3 / 31.9
PatchmatchNet [30]	43.6 / 69.4 / 53.2	27.3 / 41.7 / 32.3
PVSNet [35]	53.7 / 63.9 / 56.9	29.4 / 41.2 / 33.5
BP-MVSNet [29]	51.3 / 68.8 / 57.6	29.6 / 35.6 / 31.4
Ours	45.9 / 62.3 / 51.8	30.6 / 36.7 / 31.8

Table 2. **Results on the Tanks and Temples benchmark.** The best performing model based on F_1 score is marked as bold. Similar to Table 1. Our method performs on par with existing learning based methods on the advanced sets.

step.

To promote view-selection robustness, for each reference image, we select 6 total views from the same scene: 3 random views and 3 views sampled from the 10 best views according to BlendedMVS. Among 6 source images, we sample 1 best visibility-scoring image as visible and 2 worst visibility-scoring images as invisible. We train the model with Adam [17] and set the initial learning rate to 0.001 and the decay to 0.5 per epoch. We implemented our approach in PyTorch. We use an Nvidia RTX 3090 for training and evaluation.

5.2. ETH3D High-Res Multi-View Benchmark

We evaluate our method on the ETH3D High-res Multi-View Benchmark, which contains 17 different indoor or outdoor scenes with 6048x4032 resolution images for each scene. For evaluation, we fix the number of the source views to 10 and sample the 3 best views. We use a fixed image resolution of 1920×1280 with camera intrinsics obtained by COLMAP [27]. The system takes 13.5 seconds and uses 7693MB of peak memory for each reference image. Table 1 provides quantitative results. We show that our method

Model	Accuracy / Completeness / F1	
	Train 2CM	Train 5cm
w/o normal	62.9 / 58.0 / 54.0	81.1 / 76.7 / 75.1
w/o view sel.	75.8 / 56.7 / 64.1	89.4 / 72.9 / 79.8
w/o rcr.	75.6 / 60.9 / 66.7	89.0 / 77.9 / 82.6
Ours	76.1 / 62.2 / 67.8	90.5 / 78.8 / 83.3

Table 3. **Ablation Study on ETH3D High-Res Training Set.** We compare our original system to using 1x1 feature patches that do not take advantage of normal estimates (“w/o normal”), without pixelwise view selection, instead using the top 3-ranked source images for all pixels, and without recurrent cost regularization (“w/o rcr”). The model with the highest F_1 score is marked with bold for each threshold.

achieves comparable results to the other listed methods on the standard 2cm benchmark and the best results on the 5cm benchmark. Most learning-based methods fail to produce reasonable results on ETH3D because there are few images with wide baselines and large depth ranges. In Figure 6, we compare the inferred depth and normal maps with COLMAP [27]. From the results, we can see that our method can cover weakly textured regions, such as white walls and pillars, more completely than COLMAP [27], while still maintaining good accuracy. However, the model may fail to reconstruct reflective surfaces and large texture-less areas.

5.3. Tanks and Temples Benchmark

With the same trained model, we evaluate on the Tanks and Temples [18] intermediate and advanced benchmarks which contain 8 intermediate and 6 advanced large-scale scenes respectively. Similar to the ETH3D High-res benchmark, we fix the number of the source views to 10, sample the 3 best views, and fix the image resolution to 1920×1080 . Our method takes 12.1 seconds and uses 5801MB of peak memory for each reference image. Table 2 shows the quantitative results of the benchmark. We achieve similar results to CasMVSNet [11] and PatchmatchNet [30]. In Figure 7, we present qualitative results on the reconstructed point clouds.

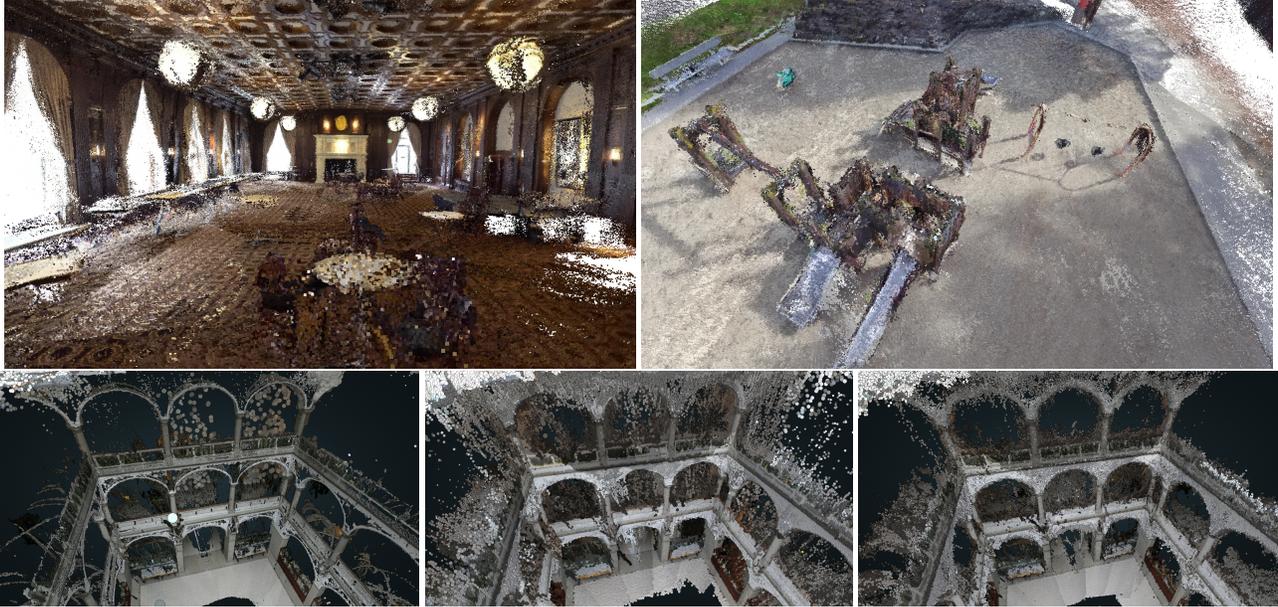


Figure 7. **Point cloud reconstruction results.** For the top row, from left to right, we show *Ballroom* and *Playground* from the Tanks and Temples benchmark [18]. For the bottom row, from left to right, we show the reconstruction results of COLMAP [27], PatchmatchNet [30] and our method in the *Exhibition Hall* from the ETH3D benchmark [28]. See the [benchmark website](#) for more reconstruction results.

We show that our method can generate complete and accurate reconstructions, which includes repeated textures such as carpet and thin structures such as poles for the swing.

5.4. Ablation Studies

Table 3 shows how each component contributes to the performance of our method.

Importance of normals: Our use of normals enables modeling oblique surfaces and provides a locally planar support region for photometric costs, which has otherwise been achieved through deformable convolution [30] or k -NN [6]. Without normal estimation for more non-frontal planar propagation and support, accuracy drops by 13.2% and completeness drops by 3.2% for the 2cm threshold on ETH3D (Table 3 “w/o normal”).

Importance of pixelwise view selection: Without pixelwise selection of the source images, the completeness of the reconstruction at the 2cm threshold drops by 5.5% and accuracy drops slightly (Table 3 “w/o view sel”). Pixelwise view selection makes better use of many source views that are partially overlapping the reference image.

Importance of recurrent cost regularization: We introduce recurrent cost regularization to aggregate confidences (i.e. feature correlations) across similar points without requiring aligned cost volumes. For comparison, we try scoring candidates using a multi-layer network based on only the feature correlations for the centered patch. With this simplification, the overall $F1$ score drops by 1.1% for the 2cm threshold (Table 3 “w/o rcr”).

Importance of argmax sampling: We tried to train using soft-argmax based candidates where we take the expectation of the normals and the depths independently. However,

we failed to train an effective model due to the initial aggregated values being clustered to the middle of the depth ranges, which limits the range of predictions. Existing works may avoid this problem by using sampling in a more restrictive way; e.g., by performing a depth search with reduced range [8] or by sampling initial points from uniformly separated bins [30]. Our reinforcement learning approach enables us to perform argmax sampling in the same way as non-learning based approaches while benefiting from learned representations.

6. Conclusion

We propose an end-to-end trainable MVS system that estimates pixelwise depths, normals, and visibilities using PatchMatch optimization. We use reinforcement learning and a decaying ϵ -greedy sampling in training to learn effectively despite using view selection and argmax sampling in inference. Our system performs well compared to the latest learning-based MVS systems, but further improvements are possible. For example, we have not yet incorporated some of the sophisticated geometric checks of ACMM [32] or post-process refinement of DeepC-MVS [19], and higher resolution processing would also yield better results. By incorporating most of the critical ideas from non-learning based methods into a learning-based framework, our work provides a promising direction for further improvements in end-to-end approaches.

Acknowledgements

We thank ONR MURI Award N00014-16-1-2007 for support in our research.

References

- [1] Awesome 3d reconstruction list. https://github.com/openMVG/awesome_3DReconstruction_list. 2
- [2] 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*, 120(2):153–168, 2016. 1
- [3] Connelly Barnes, Eli Shechtman, Adam Finkelstein, and Dan B Goldman. Patchmatch: A randomized correspondence algorithm for structural image editing. 2
- [4] Frederic Besse, Carsten Rother, Andrew Fitzgibbon, and Jan Kautz. Pmbp: Patchmatch belief propagation for correspondence field estimation. In *BMVC - Best Industrial Impact Prize award*, January 2012. 4
- [5] Michael Bleyer, Christoph Rhemann, and Carsten Rother. Patchmatch stereo-stereo matching with slanted support windows. 1, 2
- [6] Rui Chen, Songfang Han, Jing Xu, and Hao Su. Point-based multi-view stereo network. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1538–1547, 2019. 4, 8
- [7] Joseph DeGol, Mani Golparvar-Fard, and Derek Hoiem. Geometry-informed material recognition. In *CVPR*, 2016. 1
- [8] Shivam Duggal, Shenlong Wang, Wei-Chiu Ma, Rui Hu, and Raquel Urtasun. Deeppruner: Learning efficient stereo matching via differentiable patchmatch. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 4384–4393, 2019. 1, 2, 8
- [9] Y. Furukawa and C. Hernández. Multi-view stereo: A tutorial. *Found. Trends Comput. Graph. Vis.*, 9:1–148, 2015. 2
- [10] 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. 3, 4, 5, 7
- [11] Xiaodong Gu, Zhiwen Fan, Siyu Zhu, Zuoqun Dai, Feitong Tan, and Ping Tan. Cascade cost volume for high-resolution multi-view stereo and stereo matching. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020. 7
- [12] Xiaoyang Guo, Kai Yang, Wukui Yang, Xiaogang Wang, and Hongsheng Li. Group-wise correlation stereo network. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019. 1
- [13] Kevin Han, Joseph Degol, and Mani Golparvar-Fard. Geometry- and appearance-based reasoning of construction progress monitoring. *Journal of Construction Engineering and Management*, 144(2):04017110, 2018. 1
- [14] Yinlin Hu, Rui Song, and Yunsong Li. Efficient coarse-to-fine patchmatch for large displacement optical flow. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5704–5712, 2016. 2
- [15] 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
- [16] Mengqi Ji, Juergen Gall, Haitian Zheng, Yebin Liu, and Lu Fang. Surfacer-net: An end-to-end 3d neural network for multi-view stereopsis. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2307–2315, 2017. 1, 2
- [17] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In Yoshua Bengio and Yann LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015. 7
- [18] 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. 1, 6, 7, 8
- [19] Andreas Kuhn, Christian Sormann, Mattia Rossi, Oliver Erdler, and Friedrich Fraundorfer. Deepc-mvs: Deep confidence prediction for multi-view stereo reconstruction. In *2020 International Conference on 3D Vision (3DV)*, pages 404–413. IEEE, 2020. 2, 8
- [20] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2117–2125, 2017. 3
- [21] Jiangbo Lu, Hongsheng Yang, Dongbo Min, and Minh N Do. Patch match filter: Efficient edge-aware filtering meets randomized search for fast correspondence field estimation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1854–1861, 2013. 2
- [22] Bruce D. Lucas and Takeo Kanade. An iterative image registration technique with an application to stereo vision. In *Proc. International Joint Conference on Artificial Intelligence (IJCAI)*, page 674–679, 1981. 2
- [23] Keyang Luo, Tao Guan, Lili Ju, Yuesong Wang, Zhuo Chen, and Yawei Luo. Attention-aware multi-view stereo. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1590–1599, 2020. 2, 7
- [24] Mervin E. Muller. A note on a method for generating points uniformly on n-dimensional spheres. *Commun. ACM*, 2(4):19–20, Apr. 1959. 3
- [25] Henri Rebecq, Guillermo Gallego, Elias Mueggler, and Davide Scaramuzza. Emvs: Event-based multi-view stereo—3d reconstruction with an event camera in real-time. *International Journal of Computer Vision*, 126(12):1394–1414, 2018. 1
- [26] Andrea Romanoni and Matteo Matteucci. Tapa-mvs: Textureless-aware patchmatch multi-view stereo. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 10413–10422, 2019. 1, 2
- [27] Johannes Lutz Schönberger, Enliang Zheng, Marc Pollefeys, and Jan-Michael Frahm. Pixelwise View Selection for Unstructured Multi-View Stereo. In *European Conference on Computer Vision (ECCV)*, 2016. 1, 2, 4, 5, 7, 8
- [28] 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, 6, 8
- [29] C. Sormann, P. Knobelreiter, A. Kuhn, M. Rossi, T. Pock, and F. Fraundorfer. Bp-mvsnet: Belief-propagation-layers for multi-view-stereo. In *2020 International Conference on 3D Vision (3DV)*, pages 394–403, Los Alamitos, CA, USA, nov 2020. IEEE Computer Society. 7

- [30] Fangjinhua Wang, Silvano Galliani, Christoph Vogel, Pablo Speciale, and Marc Pollefeys. Patchmatchnet: Learned multi-view patchmatch stereo, 2020. [3](#), [7](#), [8](#)
- [31] Rui Wang, Martin Schworer, and Daniel Cremers. Stereo dso: Large-scale direct sparse visual odometry with stereo cameras. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3903–3911, 2017. [1](#)
- [32] 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. [1](#), [2](#), [7](#), [8](#)
- [33] Qingshan Xu and Wenbing Tao. Learning inverse depth regression for multi-view stereo with correlation cost volume. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 12508–12515, 2020. [4](#), [7](#)
- [34] Qingshan Xu and Wenbing Tao. Planar prior assisted patchmatch multi-view stereo. *AAAI Conference on Artificial Intelligence (AAAI)*, 2020. [1](#), [2](#)
- [35] Qingshan Xu and Wenbing Tao. Pvsnet: Pixelwise visibility-aware multi-view stereo network. *arXiv preprint arXiv:2007.07714*, 2020. [2](#), [7](#)
- [36] 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. [2](#), [3](#), [4](#), [5](#)
- [37] 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. [2](#), [7](#)
- [38] Yao Yao, Zixin Luo, Shiwei Li, Jingyang Zhang, Yufan Ren, Lei Zhou, Tian Fang, and Long Quan. Blendedmvs: A large-scale dataset for generalized multi-view stereo networks. *Computer Vision and Pattern Recognition (CVPR)*, 2020. [6](#)
- [39] Zehao Yu and Shenghua Gao. Fast-mvsnet: Sparse-to-dense multi-view stereo with learned propagation and gauss-newton refinement. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1949–1958, 2020. [1](#), [2](#)
- [40] Jure Žbontar and Yann LeCun. Stereo matching by training a convolutional neural network to compare image patches. *The journal of machine learning research*, 17(1):2287–2318, 2016. [1](#)
- [41] Enliang Zheng, Enrique Dunn, Vladimir Jovic, and Jan-Michael Frahm. Patchmatch based joint view selection and depthmap estimation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1510–1517, 2014. [1](#), [2](#)