3DVNet: Multi-View Depth Prediction and Volumetric Refinement

Alexander Rich Noah Stier Pradeep Sen Tobias Höllerer
{anrich, noahstier, psen, thollerer}@ucsb.edu
University of California, Santa Barbara

2021 International Conference on 3D Vision (3DV) | 978-1-6654-2688-6/21/$31.00 ©2021 IEEE | DOI: 10.1109/3DV53792.2021.00079

Abstract

We present 3DVNet, a novel multi-view stereo (MVS) depth-prediction method that combines the advantages of previous depth-based and volumetric MVS approaches. Our key idea is the use of a 3D scene-modeling network that iteratively updates a set of coarse depth predictions, resulting in highly accurate predictions which agree on the underlying scene geometry. Unlike existing depth-prediction techniques, our method uses a volumetric 3D convolutional neural network (CNN) that operates in world space on all depth maps jointly. The network can therefore learn meaningful scene-level priors. Furthermore, unlike existing volumetric MVS techniques, our 3D CNN operates on a feature-augmented point cloud, allowing for effective aggregation of multi-view information and flexible iterative refinement of depth maps. Experimental results show our method exceeds state-of-the-art accuracy in both depth prediction and 3D reconstruction metrics on the ScanNet dataset, as well as a selection of scenes from the TUM-RGBD and ICL-NUIM datasets. This shows that our method is both effective and generalizes to new settings.

1. Introduction

Multi-view stereo (MVS) is a central problem in computer vision with applications from augmented reality to autonomous navigation. In MVS, the goal is to reconstruct a scene using only posed RGB images as input. This reconstruction can take many forms, from voxelized occupancy or truncated signed distance fields (TSDFs), to per-frame depth prediction, the focus of this paper. In recent years, MVS methods based on deep learning [2, 6, 11, 12, 17, 18, 22, 24, 26, 29, 30, 31, 32] have surpassed traditional MVS methods [9, 21] on numerous benchmark datasets [5, 13, 15]. In this work, we consider these methods as falling into two categories, depth estimation and volumetric reconstruction, each with advantages and disadvantages.

The most recent learning methods in depth estimation use deep features to perform dense multi-view matching robust to large environmental lighting changes and textureless or specular surfaces, among other things. These methods take advantage of well researched multi-view aggregation techniques and the flexibility of depth as an output modality. They formulate explicit multi-view matching costs and include iterative refinement layers in which a network predicts a small depth offset between an initial prediction and the ground truth depth map [2, 32]. While these techniques have been successful for depth prediction, most are constrained to making independent, per-frame predictions. This results in predictions that do not agree on the underlying 3D geometry of the scene. Those that do make joint predictions across multiple frames use either regularization constraints [11] or recurrent neural networks (RNNs) [6] to encourage frames close in pose space to make similar predictions. However, these methods do not directly operate on a unified 3D scene representation, and their resulting reconstructions lack global coherence (see Fig. 1).

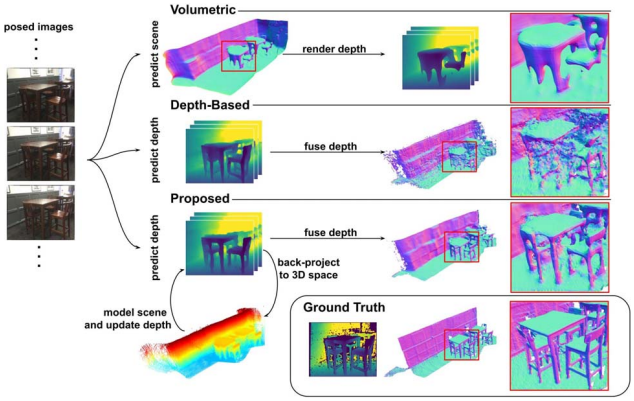

Figure 1: Volumetric methods lack local detail while depth-based methods lack global coherence. Our method cyclically predicts depth, back-projects into 3D space, volumetrically models geometry, and updates all depth predictions to match, resulting in local detail *and* global coherence.

Meanwhile, volumetric techniques operate directly on a unified 3D scene representation by back-projecting and aggregating 2D features into a 3D voxel grid and using a 3D

https://github.com/alexrich021/3dvnet

978-1-6654-2688-6/21/$31.00 ©2021 IEEE
DOI 10.1109/3DV53792.2021.00079

convolutional neural network (CNN) to regress a voxelized parameter, often a TSDF. These methods benefit from the use of 3D CNNs and naturally produce highly coherent 3D reconstructions and accurate depth predictions. However, they do not explicitly formulate a multi-view matching cost like depth-based methods, generally averaging deep features from different views to populate the 3D voxel grid. This results in overly-smooth output meshes (see Fig. 1).

In this paper, we propose *3DVNet*, an end-to-end differentiable method for learned multi-view depth prediction that leverages the advantages of both volumetric scene modeling and depth-based multi-view matching and refinement. The key idea behind our method is the use of a 3D scene-modeling network which outputs a multi-scale volumetric encoding of the scene. This encoding is used with a modified PointFlow algorithm [2] to iteratively update a set of initial coarse depth predictions, resulting in predictions that agree on the underlying scene geometry.

Our 3D network operates on all depth predictions at once, and extracts meaningful, scene-level priors similar to volumetric MVS methods. However, the 3D network operates on features aggregated using depth-based multi-view matching and can be used iteratively to update depth maps. In this way, we combine the advantages of the two separate classes of techniques. Because of this, 3DVNet exceeds state-of-the-art results on ScanNet [5] in nearly all depth map prediction *and* 3D reconstruction metrics when compared with the current best depth and volumetric baselines. Furthermore, we show our method generalizes to other real and synthetic datasets [10, 23], again exceeding the best results on nearly all metrics. Our contributions are as follows:

1. We present a 3D scene-modeling network which outputs a volumetric scene encoding, and show its effectiveness for iterative depth residual prediction.

2. We modify PointFlow [2], an existing method for depth map residual predictions, to use our volumetric scene encoding.

3. We design 3DVNet, a full MVS pipeline, using our 3D scene-modeling network and PointFlow refinement.

2. Related Works

We cover MVS methods using deep learning, categorizing them as either depth-prediction methods or volumetric methods. Our method falls into the first category, but is very much inspired by volumetric techniques.

Depth-Prediction MVS Methods: With some notable exceptions [22, 28], nearly all depth-prediction methods follow a similar paradigm: (1) they construct a plane sweep cost volume on a reference image's camera frustum, (2) they fill the volume with deep features using a cost function that operates on source and reference image features, (3) they use a network to predict depth from this cost volume. Most methods differ in their cost metric used to construct the volume. Many cost metrics exist, including per-channel variance of deep features [29, 30], learned aggregation using a network [17, 31], concatenation of deep features [12], the dot product of deep features [6], and absolute intensity difference of raw image RGB values [11, 26]. We find per-channel variance [29] to be the most commonly used cost metric, and adopt it in our system.

The choice of cost aggregation method results in either a vectorized matching cost and thus a 4D cost volume [2, 12, 17, 29, 30, 31, 32] or a scalar matching cost and thus a 3D cost volume [6, 11, 26]. Methods with 4D cost volumes generally require 3D networks for processing, while 3D cost volumes can be processed with a 2D U-Net-style [20] encoder-decoder architecture. Some methods operate on the deep features at the bottleneck of this U-Net to make joint depth predictions for all N frames or a subset of frames in a given scene. This is similar to our proposed method, and we highlight the differences.

GPMVS [11] uses a Gaussian Process (GP) constraint conditioned on pose distance to regularize these deep features. This GP constraint only operates on deep features and assumes Gaussian priors. In contrast, we directly *learn* priors from predicted depth maps and explicitly predict depth residuals to modify depth maps to match. DV-MVS [6] introduces an RNN to propagate information from the deep features in frame $t-1$ to frame t given an ordered sequence of frames. While they do propagate this information in a geometrically plausible way, the RNN operates only on deep features similar to GPMVS. Furthermore, the RNN never considers all frames jointly like our method.

Similar to our method, some networks iteratively predict a residual to refine an initial depth prediction [2, 32]. We specifically highlight Point-MVSNet [2], which introduces PointFlow, a point cloud learning method for residual prediction. Our method is very much inspired by this work. We briefly describe the differences.

In their work, they operate on a point cloud back-projected from a *single* depth map and augmented with additional points. Features are extracted from this point cloud using point cloud learning techniques and used in their PointFlow module for residual prediction. Crucially, these features do not come from a unified 3D representation of the scene. Thus the residual prediction is only conditioned on information local to the individual depth prediction and not global scene information. In contrast, our variation of PointFlow uses our volumetric scene model to condition residual prediction on information from *all* depth maps. For an in depth discussion of differences, see Sec. 3.2.

Volumetric MVS Methods: In volumetric MVS, the goal is to directly regress a global volumetric representation of the scene, generally a TSDF volume. We highlight

two methods which inspired our work. Atlas [18] back-projects rays of deep features extracted from images into a global voxel grid, pools features from multiple views using a running average, then directly regress a TSDF in a coarse-to-fine fashion using a 3D U-Net. NeuralRecon [24] improves on the memory consumption and run-time of Atlas by reconstructing local fragments using the most recent keyframes, then fusing the local fragments to a global volume using an RNN. The reconstructions these methods produce are pleasing. However, both construct feature volumes using averaging in a single forward pass, which we believe is non-optimal. In contrast, our depth-based method allows us to construct a feature volume using multi-view matching features and perform iterative refinement.

3. Methods

Our method takes as input N images, denoted $\{\mathbf{I}_n\}$, $n = 1, \ldots, N$ with corresponding known extrinsic and intrinsic camera parameters. Our goal is to predict N depth maps $\{\mathbf{D}_n\}$ corresponding to the N images. As a pre-processing step, we define for every image $\mathbf{I}_n$ a set of M indices $\{s_1, \ldots, s_M\}$ pointing to which images to use as source images for depth prediction, and append the reference index to form the set $\mathbf{S}_n = \{n, s_1, \ldots, s_M\}$.

Our pipeline is as follows. First, a small depth-prediction network is used to independently predict initial coarse depth maps $\{\mathbf{D}_n^0\}$ for every frame $\{\mathbf{I}_n\}$ using extracted image features $\{\mathbf{F}_n\}$ (Sec. 3.3). Second, we back-project our N initial depth maps to form a joint point cloud $\mathbf{X} \subset \mathbb{R}^3$ (Sec 3.1). Because each point $\mathbf{p} \in \mathbf{X}$ is associated with one depth map $\mathbf{D}_n^0$ that has associated feature maps $\{\mathbf{F}_s : s \in \mathbf{S}_n\}$, we can augment it with a multi-view matching feature aggregated from those feature maps. Third, our 3D scene-modeling network takes as input this feature-rich point cloud and outputs a multi-scale scene encoding $\mathbf{V}_1, \mathbf{V}_2, \mathbf{V}_3$ (Sec. 3.1). Fourth, we update each depth map to match this scene encoding using a modified PointFlow algorithm, resulting in highly coherent depth maps and thus highly coherent reconstructions (Sec. 3.2). Steps 2-4 can be run in a nested for-loop, with steps 2 and 3 run in the outer loop to generate updated scene models with the current depth maps and step 4 run in the inner loop to refine depth maps with the current scene model. We denote the updated depth map after l_o outer loop iterations of scene modeling and l_i inner loop iterations of updating as $\mathbf{D}_n^{(l_o, l_i)}$. Finally, we upsample the resulting refined depth maps to the size of the original image in a coarse-to-fine manner, guided by deep features and the original image, to arrive at final predictions $\{\mathbf{D}_n\}$ for every image $\{\mathbf{I}_n\}$ (Sec. 3.3).

3.1. 3D Scene Modeling

A visualization of our 3D scene modeling method is given in the upper half of Fig. 2. As stated previously, our 3D scene-modeling network operates on a feature rich point cloud back-projected from $\{\mathbf{D}_n^0\}$ or subsequent updated depth maps. To process this point cloud, we adopt a voxelize-then-extract approach. We first generate a sparse 3D grid of voxels, culling voxels that do not contain depth points. To avoid losing granular information of the point cloud, we generate a deep feature for each voxel using a per-voxel PointNet [1]. The PointNet inputs are the features of each depth point in the voxel as well as the 3D offset of that point to the voxel center. Finally, we run a 3D U-Net [20] on the resulting voxelized feature volume and extract intermediate outputs at multiple resolutions. By nature of construction, this U-Net learns meaningful, scene-level priors. The result is a multi-scale, volumetric scene encoding.

Point Cloud Formation: We form our point cloud $\mathbf{X} \subset \mathbb{R}^3$ by back-projecting all depth pixels in all N depth maps. For our multi-view matching feature associated with each point $\mathbf{p} \in \mathbf{X}$, we follow existing work [2, 29] and use per-channel variance aggregation using the reference and source feature maps associated with each depth pixel. For $\mathbf{p} \in \mathbf{X}$, given that $\mathbf{p}$ belongs to depth map $\mathbf{D}_n^0$, the equation for variance feature $\boldsymbol{\sigma}^2(\mathbf{p})$, applied per-channel, is:

$$\boldsymbol{\sigma}^2(\mathbf{p}) = \frac{1}{|\mathbf{S}_n|} \sum_{s \in \mathbf{S}_n} \left(\mathbf{F}_s(\hat{\mathbf{p}}_s) - \overline{\mathbf{F}_*(\hat{\mathbf{p}}_*)} \right)^2 \qquad (1)$$

where $\hat{\mathbf{p}}_s$ is the projection of $\mathbf{p}$ to feature map $\mathbf{F}_s$, $\underline{\mathbf{F}_s(\hat{\mathbf{p}}_s)}$ is the bilinear interpolation of $\mathbf{F}_s$ to point $\hat{\mathbf{p}}_s$, and $\overline{\mathbf{F}_*(\hat{\mathbf{p}}_*)}$ is the average interpolated feature over all indices $s \in \mathbf{S}_n$. Intuitively, if $\mathbf{p}$ lies on a surface it is more likely to have low variance in most feature channels in $\boldsymbol{\sigma}^2(\mathbf{p})$ while if it doesn't lie on a surface the variance will likely be high.

Point Cloud Voxelization: To form our initial feature volume, we regularly sample an initial 3D grid of points $\mathbf{C}$ every $r = 8$ cm within the axis-aligned bounding box of point cloud $\mathbf{X}$ and define the voxel associated with each grid point $\mathbf{c} \in \mathbf{C}$ as the 8 cm^3 cube with center $\mathbf{c}$. We denote the set of depth points that fall within a voxel with center $\mathbf{c} \in \mathbf{C}$ as $v(\mathbf{c}) = \{\mathbf{p} \in \mathbf{X} : ||\mathbf{c} - \mathbf{p}||_\infty <= \frac{r}{2}\}$. We sparsify this grid by discarding $\mathbf{c} \in \mathbf{C}$ if no depth points lie within the associated voxel, denoting this set of grid coordinates as $\hat{\mathbf{C}} = \{\mathbf{c} \in \mathbf{C} : v(\mathbf{c}) \neq \emptyset\}$. For $\mathbf{c} \in \hat{\mathbf{C}}$, we produce a feature for the associated voxel using PointNet [1] with max pooling. The PointNet feature for each voxel is defined as:

$$\mathbf{f}_v(\mathbf{c}) = \underset{\mathbf{p} \in v(\mathbf{c})}{\triangle} h_\theta \left(\text{concat} \left[\mathbf{p} - \mathbf{c}, \boldsymbol{\sigma}^2(\mathbf{p}) \right] \right) \qquad (2)$$

where h_θ is a learnable multi-layer perceptron (MLP), concat $[\mathbf{q}, \mathbf{f}]$ indicates concatenation of the 3D coordinates $\mathbf{q}$ with the feature channel of $\mathbf{f}$ to form a feature with 3 additional channels, and $\triangle$ is the channel-wise max pooling operation. The result of this stage is a sparse feature volume $\mathbf{V}_0$ with features given by Eq. 2 and coordinates $\hat{\mathbf{C}}$.

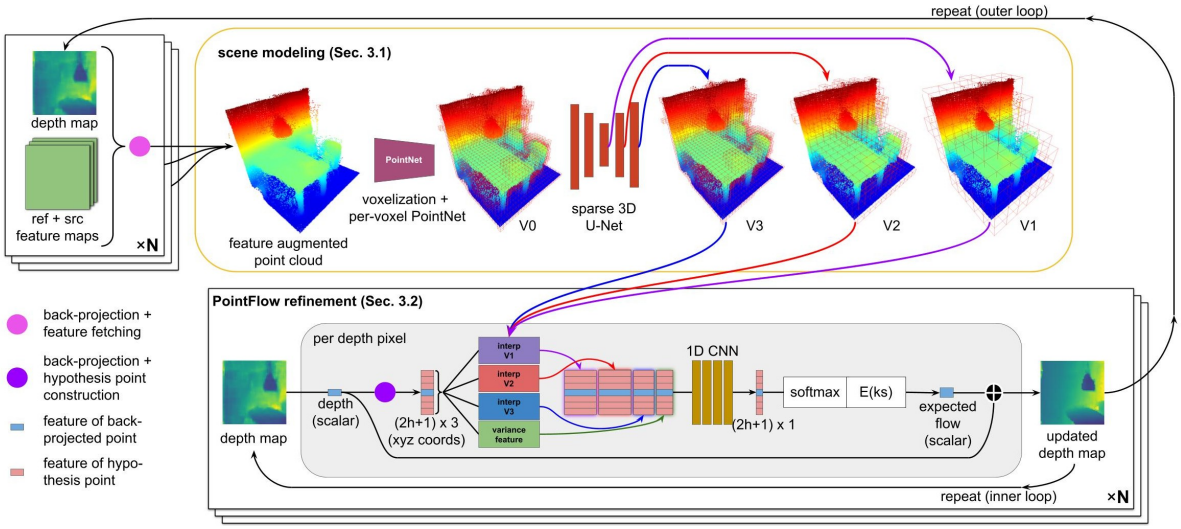

Figure 2: Our novel 3D scene modeling and refinement method first constructs a multi-scale volumetric scene encoding from a set of N input depth maps with corresponding feature maps. It then use that encoding in a variation of the PointFlow algorithm [2] to predict a residual for each of the N depth maps. The full method can be run in a nested for-loop fashion, predicting multiple residuals per depth map in the inner loop and running scene modeling in the outer loop.

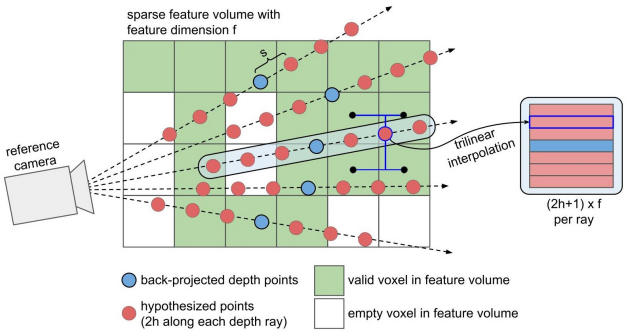

Figure 3: Diagram of standard PointFlow hypothesis point construction and our proposed feature generation, shown in 2D for simplicity. Feature volume in diagram corresponds to a single scale of our multi-scale scene encoding. Our key change from the original formulation is to generate hypothesis point features by trilinear interpolation of our volumetric scene encoding rather than edge convolutions on the point cloud from a single back-projected depth map.

Multi-Scale 3D Feature Extraction: In this stage, we use a sparse 3D U-Net to model the underlying scene geometry. We use a basic U-Net architecture with skip connections. Group normalization is used throughout. See supplementary material for a more detailed description of our architecture. Our sparse U-Net takes as input sparse feature volume $\mathbf{V}_0$. From intermediate outputs of the U-Net, we extract three scales of feature volumes $\mathbf{V}_1$, $\mathbf{V}_2$, $\mathbf{V}_3$ with a voxel edge length of $4r = 32$ cm, $2r = 16$ cm, and $r = 8$ cm, respectively, describing the scene. In this way, we extract a rich, multi-scale, volumetric encoding of the scene.

3.2. PointFlow-Based Refinement

In this stage, we use our multi-scale scene encoding $\mathbf{V}_1, \mathbf{V}_2, \mathbf{V}_3$ from the previous stage in a variation of the PointFlow algorithm proposed by Chen *et al.* [2]. The goal is to refine our predicted depth maps to match our scene model by predicting a residual for each depth pixel. We briefly review the core components of PointFlow and the intuition behind our proposed change.

In PointFlow, a set of points called *hypothesis points* are constructed at regular intervals along a depth ray, centered about the depth prediction associated with the given depth ray. The blue and red points in Fig. 3 illustrate this. Features are generated for the hypothesis points. Then, a network processes these features and outputs a probability score for every point indicating confidence the given point is at the correct depth. Finally, the expected offset is calculated using these probabilities and added to the original depth prediction. Our key innovation is the use of our multi-scale scene encoding to generate the hypothesis point features.

In the original PointFlow, hypothesis points are constructed for a *single* depth map, augmented with features using Eq. 1, and aggregated into a point cloud. Note this point cloud is strictly different from our point cloud as (1) it is produced using a *single* depth map, and (2) it includes hypothesis points. Features are generated for each point using edge convolutions [27] on the k-Nearest-Neighbor (kNN) graph. Crucially, these edge convolutions never operate on a unified 3D scene representation in the original PointFlow. This prevents the offset predictions from learning global information, which we believe is a critical step for depth residual prediction. Furthermore, because of the required

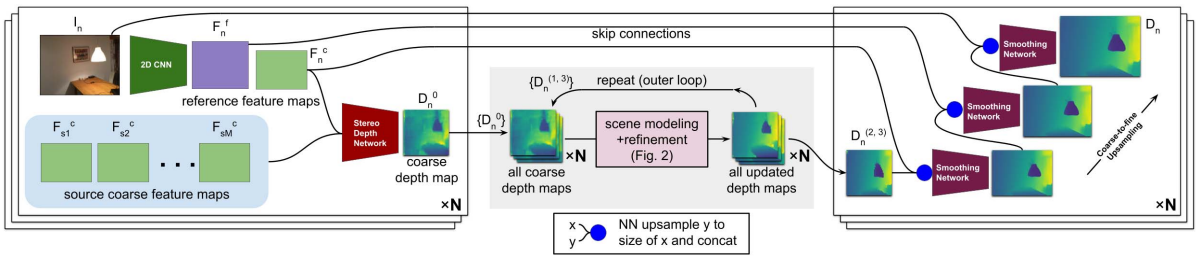

Figure 4: Overview of the full 3DVNet pipeline. See Secs. 3.1 and 3.2 for a description of our scene modeling and refinement.

kNN search, this formulation cannot scale to process a joint point cloud from an arbitrary number of depth maps, therefore preventing it from scaling to learn global information.

Inspired by convolutional occupancy networks [19] and IFNets [3], we instead generate hypothesis features by interpolating each scale of our multi-scale scene encoding (see Fig. 3). With this key change, we use powerful scene-level priors in our offset prediction conditioned on all N depth predictions for a given scene. Furthermore, by using the same encoding to update all N depth predictions, we encourage global consistency of predictions. We now describe in detail our variation of the PointFlow method (see Figs. 2 and 3), using notation similar to the original paper.

Hypothesis Point Construction: For a given back-projected depth pixel $\mathbf{p}$ from depth map $\mathbf{D}_n$, we generate $2h+1$ point hypotheses $\{\tilde{\mathbf{p}}_k\}$:

$$\tilde{\mathbf{p}}_k = \mathbf{p} + ks\mathbf{t}, \quad k = -h, \ldots, h \tag{3}$$

where $\mathbf{t}$ is the normalized reference camera direction of $\mathbf{D}_n$, and s is the displacement step size.

Feature Generation: We generate a multi-scale feature for each hypothesis point $\tilde{\mathbf{p}}_k$ using trilinear interpolation to point $\tilde{\mathbf{p}}_k$ of our sparse features volumes $\mathbf{V}_1, \mathbf{V}_2, \mathbf{V}_3$, using 0s where features are not defined:

$$\mathbf{f}_i(\tilde{\mathbf{p}}_k) = \text{sparse\_interp}(\mathbf{V}_i, \tilde{\mathbf{p}}_k), \quad i = 1, 2, 3 \tag{4}$$

Next, we generate a variance feature $\boldsymbol{\sigma}^2(\tilde{\mathbf{p}}_k)$ for hypothesis point $\tilde{\mathbf{p}}_k$ using Eq. 1. The final feature for a hypothesis point is the channel-wise concatenation of these features:

$$\mathbf{f}_k(\tilde{\mathbf{p}}_k) = \text{concat}\left[\mathbf{f}_1(\tilde{\mathbf{p}}_k), \mathbf{f}_2(\tilde{\mathbf{p}}_k), \mathbf{f}_3(\tilde{\mathbf{p}}_k), \boldsymbol{\sigma}^2(\tilde{\mathbf{p}}_k)\right] \tag{5}$$

We stack our $2h+1$ point-hypothesis features to form a 2D feature $\mathbf{H} \in \mathbb{R}^{(2h+1)\times c}$, where c is the sum of the dimensions of our variance and scene encoding features.

Offset Prediction: We apply a 4 layer 1D CNN followed by a softmax function to predict a probability scalar for each point-wise entry in $\mathbf{H}$. The predicted displacement of point $\mathbf{p}$ is then as follows:

$$\Delta d_p = \mathbb{E}(ks) = \sum_{k=-h}^{h} ks \times \text{Prob}(\tilde{\mathbf{p}}_k) \tag{6}$$

The updated depth for each depth map is the depth of point $\mathbf{p} + \mathbf{t}\Delta d_p$ with respect to the camera associated with $\mathbf{D}_n$.

3.3. Bringing It All Together: 3DVNet

In this section, we describe our full depth-prediction pipeline using our multi-scale volumetric scene modeling and PointFlow-based refinement, which we name 3DVNet (see Fig. 4). Our pipeline consists of (1) initial feature extraction and depth prediction, (2) scene modeling and refinement, and (3) upsampling of our refined depth map to the size of the original image. The scene modeling and refinement is done in a nested for-loop fashion, extracting a scene model in the outer loop and iteratively refining the depth predictions using that scene model in the inner loop. We fix the input image size of 3DVNet to 320×256.

2D Feature Extraction: For our 2D feature extraction, we adopt the approach of Düzçeker *et al.* [6], and use a 32 channel feature pyramid network (FPN) [16] constructed on a MnasNet [25] backbone to extract coarse and fine resolution feature maps of size 80×64 and 160×128 respectively. For every image $\mathbf{I}_n$, we denote these $\mathbf{F}_n^c$ and $\mathbf{F}_n^f$.

MVSNet Prediction: For the coarse depth prediction of image $\mathbf{I}_n$, we use a small MVSNet [29] using the reference and source coarse feature maps $\{\mathbf{F}_s^c : s \in \mathbf{S}_n\}$ to predict an initial coarse depth $\mathbf{D}_n^0$. Our cost volume is constructed using traditional plane sweep stereo with $L = 96$ depth hypotheses sampled uniformly at intervals of 5 cm starting at 50 cm. Similar to Yu and Gao [32], our predicted depth map is spatially sparse compared to feature map $\mathbf{F}_n^c$. We fix our coarse depth map prediction size to 56×56.

Nested For-Loop Refinement: We denote the updated depths after scene-modeling iteration l_o and PointFlow iteration l_i as $\{\mathbf{D}_n^{(l_o,l_i)}\}$. We use initial depth predictions $\{\mathbf{D}_n^0\}$ and coarse feature maps $\{\mathbf{F}_n^c\}$ to generate multi-scale scene encoding $\mathbf{V}_1$, $\mathbf{V}_2$, $\mathbf{V}_3$. We then run PointFlow refinement three times with displacement step size $s = 5$ cm, 5 cm, and 2.5 cm and $h = 3$ to get updated depths $\{\mathbf{D}_n^{(1,3)}\}$. In early experiments, we found two iterations at 5 cm to be helpful. We re-generate our scene encoding using updated depths $\{\mathbf{D}_n^{(1,3)}\}$ and coarse feature maps $\{\mathbf{F}_n^c\}$. We then run PointFlow three more times with step sizes $s = 5$ cm, 5 cm, and 2.5 cm and $h = 3$ to get updated depths $\{\mathbf{D}_n^{(2,3)}\}$.

We find our depth maps converge at this point.

Coarse-to-Fine Upsampling: In this stage, we upsample each refined depth prediction $\mathbf{D}_n^{(2,3)}$ to the size of image $\mathbf{I}_n$. We find PointFlow refinement does not remove interpolation artifacts, as this generally requires predicting large offsets across depth boundaries. We outline a simple, coarse-to-fine method for upsampling while removing artifacts. See the right section of Fig. 4. At each step, we upsample the current depth prediction using nearest-neighbor interpolation to the size of the next-largest feature map and concatenate, using the original image $\mathbf{I}_n$ in the final step. We then pass the concatenated feature map and depth through a smoothing network. We use a version of the propagation network proposed by Yu and Gao [32]. For every pixel $\mathbf{p}$ in depth map $\mathbf{D}$, the smoothed depth $\tilde{\mathbf{D}}$ is a weighted sum of $\mathbf{D}$ in the 3×3 neighborhood about $\mathbf{p}$:

$$\tilde{\mathbf{D}}(\mathbf{p}) = \sum_{\mathbf{q} \in [-1,0,1]^2} g_\theta\left(\mathbf{p}, \mathbf{q}\right) \mathbf{D}(\mathbf{p} + \mathbf{q}) \tag{7}$$

where g_θ is a 4 layer CNN that takes as input the concatenated feature and depth map and outputs 9 weights for every pixel $\mathbf{p}$, and $g_\theta\left(\mathbf{p}, \mathbf{q}\right)$ indexes those weights for the pixel $\mathbf{p}$. A softmax function is applied to the weights for normalization. We apply this coarse-to-fine upsampling to every refined depth map $\{\mathbf{D}_n^{(2,3)}\}$ to arrive at a final depth prediction $\{\mathbf{D}_n\}$ for every input image $\{\mathbf{I}_n\}$.

4. Experiments

4.1. Implementation and Training Details

Libraries: Our model is implemented in PyTorch using PyTorch Lightning [7] and PyTorch Geometric [8]. We use Minkowski Engine [4] as our sparse tensor library. We use Open3D [33] for both visualization and evaluation.

Training Parameters: We train our network on a single NVIDIA RTX 3090 GPU. Our network is trained end-to-end with a mini-batch size of 2. Each mini-batch consists of 7 images for depth prediction. For our loss function, we accumulate the average L_1 error between ground truth and predicted depth maps, appropriately downsampling the ground truth depth map to the correct resolution, for all predicted, refined, and upsampled depth map at every stage in our pipeline. Additionally, we employ random geometric scale augmentation with a factor selected between 0.9 to 1.1 and random rotation about the gravitational axis.

We first train with the pre-trained MnasNet backbone frozen using the Adam optimizer [14] with an initial learning rate of 10^{-3} which is divided by 10 every 100 epochs ($\sim$1.5k iterations), to convergence ($\sim$1.8k iterations). We unfreeze the MnasNet backbone and finetune the entire network using Adam and an initial learning rate of 10^{-4} that is halved every 50 epochs to convergence ($\sim$1.8k iterations).

4.2. Datasets, Baselines, Metrics, and Protocols

Datasets: To train and validate our model, we use the ScanNet [5] official training and validation splits. For our main comparison experiment, we use the ScanNet official test set, which consists of 100 test scenes in a variety of indoor settings. To evaluate the generalization ability of our model, we select 10 sequences from TUM-RGBD [23], and 4 sequences from ICL-NUIM [10] for comparison.

Baselines: We compare our method to seven state of the art baselines: Point-MVSNet (PMVS) [2], Fast-MVSNet (FMVS) [32], DeepVideoMVS pair/fusion networks (DVMVS pair/fusion) [6], GPMVS batched [11], Atlas [18], and NeuralRecon [24]. The first five baselines are depth-prediction methods while the last two are volumetric methods. Of these, we consider GPMVS and Atlas the most relevant depth and volumetric methods respectively, as both use information from all frames simultaneously during inference. We use the ScanNet training scenes to fintetune methods not trained on ScanNet [2, 11, 32]. We report both the finetuned and pretrained results, denoted with and without "FT". To account for range differences between the DTU dataset [13] and ScanNet, we use our model's plane sweep parameters with PMVS and FMVS.

Metrics: We use the 2D and 3D metrics presented by Murez *et al*. [18] for evaluation. See supplementary for definitions. Amongst these metrics, we consider Abs-rel, Abs-diff, and the first inlier ratio metric $\delta < 1.25$ as the most suitable 2D metrics for measuring depth prediction quality, and F-score as the most suitable 3D metric for measuring 3D reconstruction quality. Following Düzçeker *et al*. [6], we only consider ground truth depth values greater than 50 cm to account for some methods not being able to predict smaller depth. We note F-score, Precision, and Recall are calculated per-scene and then averaged across all the scenes. This results in a different F-score than when calculating from the averaged Precision and Recall reported.

Protocols: For depth-based methods, we fuse predicted depths using the standard multi-view consistency based point cloud fusion. Based on results on validation sets, we modify the implementation of Galliani *et al*. [9] to use *depth*-based multi-view consistency check, rather than a *disparity*-based check (see Sec. 3.3 of the supplementary materials). For volumetric methods, we use marching cubes to extract a mesh from the predicted TSDF. Following Murez *et al*. [18], we trim the meshes to remove geometry not observed in the ground truth camera frustums. Additionally, ScanNet ground truth meshes often contain holes in observed regions. We mask out these holes for all baselines to avoid false penalization. All meshes are single layer to match ScanNet ground truth as noted by Sun *et al*. [24].

We use the DVMVS keyframe selection. For depth-based methods, we use each keyframe as a reference image for depth prediction. We use the 2 previous and 2

	PMVS	PMVS (FT)	FMVS	FMVS (FT)	DVMVS pair	DVMVS fusion	GPMVS	GPMVS (FT)	Atlas	Neural-Recon	Ours
SCANNET											
Abs-rel ↓	0.389	0.085	0.274	0.084	0.069	0.061	0.121	0.062	0.062	0.063	**0.040**
Abs-diff ↓	0.668	0.168	0.444	0.165	0.142	0.127	0.214	0.124	0.116	0.099	**0.079**
Abs-inv ↓	0.148	0.048	0.145	0.050	0.044	0.038	0.066	0.039	0.044	0.039	**0.026**
Sq-rel ↓	0.798	0.046	0.463	0.045	0.026	0.021	0.860	0.022	0.040	0.039	**0.015**
RMSE ↓	1.051	0.267	0.776	0.267	0.220	0.200	0.339	0.199	0.238	0.206	**0.154**
$\delta < 1.25$ ↑	0.630	0.922	0.732	0.922	0.949	0.963	0.890	0.960	0.935	0.948	**0.975**
$\delta < 1.25^2$ ↑	0.768	0.981	0.857	0.979	0.989	**0.992**	0.971	**0.992**	0.971	0.976	**0.992**
$\delta < 1.25^3$ ↑	0.859	0.994	0.915	0.993	0.997	0.997	0.990	**0.998**	0.985	0.989	0.997
Acc ↓	0.093	**0.039**	0.059	0.043	0.059	0.067	0.077	0.057	0.078	0.058	0.051
Comp ↓	0.303	0.256	0.184	0.212	0.145	0.128	0.150	0.111	0.097	0.108	**0.075**
Prec ↑	0.651	**0.738**	0.570	0.707	0.595	0.557	0.486	0.604	0.607	0.636	0.715
Rec ↑	0.317	0.433	0.486	0.454	0.489	0.504	0.453	0.565	0.546	0.509	**0.625**
F-score ↑	0.409	0.529	0.511	0.541	0.524	0.520	0.459	0.574	0.573	0.564	**0.665**
TUM-RGBD											
Abs-rel ↓	0.318	0.111	0.273	0.113	0.117	0.095	0.102	0.093	0.163	0.106	**0.076**
Abs-diff ↓	0.642	0.275	0.573	0.281	0.339	0.273	0.243	0.239	0.404	**0.167**	0.210
$\delta < 1.25$ ↑	0.662	0.858	0.694	0.851	0.838	0.886	0.874	0.891	0.816	**0.912**	**0.912**
F-score ↑	0.115	0.145	0.150	0.154	0.141	0.162	0.157	0.170	0.129	0.117	**0.181**
ICL-NUIM											
Abs-rel ↓	0.614	0.107	0.303	0.095	0.106	0.114	0.107	0.066	0.110	0.123	**0.050**
Abs-diff ↓	1.469	0.262	0.707	0.245	0.278	0.322	0.290	0.176	0.332	0.303	**0.120**
$\delta < 1.25$ ↑	0.311	0.877	0.659	0.894	0.878	0.847	0.855	0.965	0.833	0.709	**0.980**
F-score ↑	0.064	0.144	0.382	0.246	0.173	0.150	0.241	0.323	0.194	0.055	**0.440**

Table 1: Metrics for three datasets (ScanNet, TUM-RGBD, and ICL-NUIM). Bold indicates best performing method, underline the second best. White rows indicate 2D depth metrics while gray rows indicate 3D metrics. Vertical lines separate depth-based methods, volumetric methods, and our method. "FT" denotes method was finetuned on ScanNet. Our method outperforms all other baseline methods by a wide margin on most metrics.

next keyframes as source images (4 source images total). For depth-based methods, we resize the output depth map to 640×480 using nearest-neighbor interpolation. For volumetric methods, we use the predicted mesh to render 640×480 depth maps for each keyframe.

4.3. Evaluation Results and Discussion

See Tab. 1 for 2D depth and 3D geometry metrics on all datasets. Our method outperforms all baselines by a wide margin on most metrics. Notably, our Abs-rel error on ScanNet is 0.021 less than the DVMVS fusion, the second best method, while the Abs-rel of the third, fourth, and fifth best methods are all within 0.002 of DVMVS fusion. Similarly, Our ScanNet F-score is 0.09 more than GPMVS (FT), the second best method, while the F-score is within 0.001 of GPMVS (FT) for Atlas, the third best method. This demonstrates the significant quantitative increase in both depth and reconstruction metrics of our method. Results on additional datasets show the strong generalization ability of our model.

We include qualitative results on ScanNet. See Figs. 5 and 6. See Sec. 4 of the supplementary materials for additional qualitative results. Our depth maps are visually pleasing, with clearly defined edges. They are comparable in quality to those of GPMVS and DVMVS fusion while being quantitatively more accurate. Our reconstructions are coherent like volumetric methods, without the noise present in other depth-based reconstructions, which we believe is a result of our volumetric scene encoding and refinement.

We do note one benefit of Atlas is its ability to fill large unobserved holes. Though not reflected in the metrics, this leads to qualitatively more complete reconstructions. Our system relies on depth maps and thus cannot do this as designed. However, as a result of averaging across image features, Atlas produces meshes that are overly smooth and lack detail. In contrast, our reconstructions contain sharper, better defined features than purely volumetric methods. Finally, we note our system cannot naturally be run in an online fashion, requiring availability of all frames prior to use.

4.4. Ablation and Additional Studies

Does Iterative Refinement Help? We study the effect of each inner and outer loop iteration of our depth refinement. See Tab. 2. We exceed state-of-the-art metrics after 2 iterations. 3 additional iterations add continued improvement, confirming the effectiveness of iterative refinement. By 5 iterations, our metrics have converged, with depth sta-

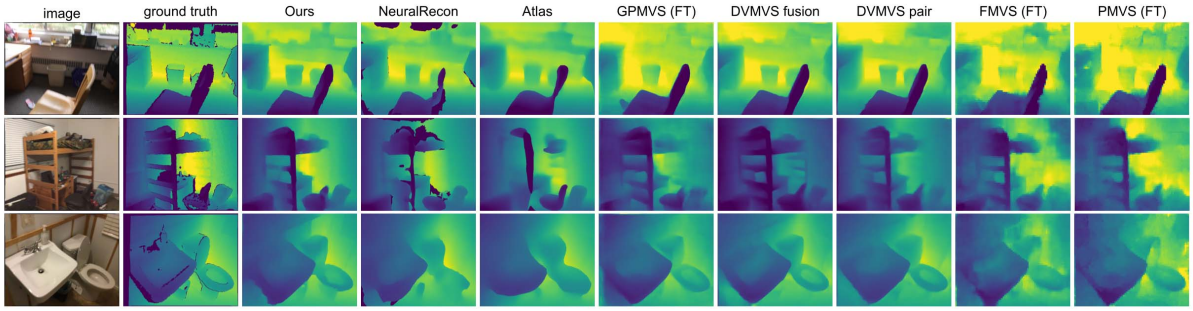

Figure 5: Qualitative depth results on ScanNet. Our method produces sharp details with well defined object boundaries.

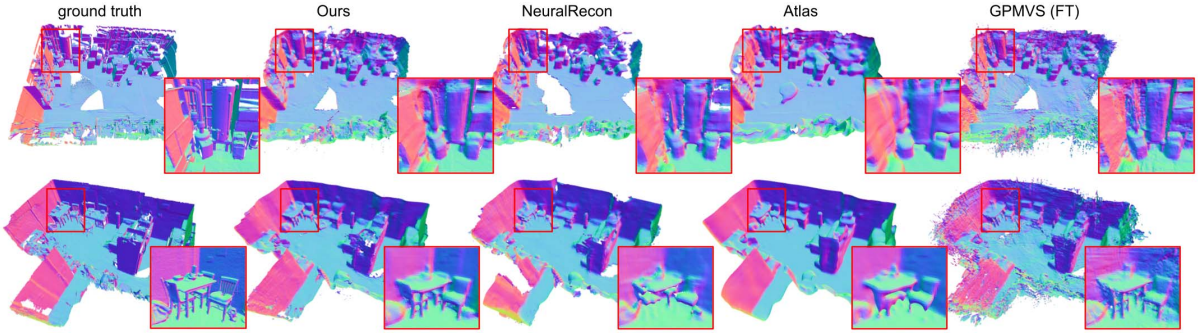

Figure 6: Qualitative reconstruction results on ScanNet for the four best-performing methods. Our technique produces globally coherent reconstructions like purely volumetric methods while containing the local detail of depth-based methods.

l_o	l_i	Abs-rel	Abs-diff	$\delta < 1.25$	F-score
0	0	0.070	0.137	0.949	0.559
1	1	0.050	0.100	0.965	0.651
1	2	0.044	0.088	0.971	0.661
1	3	0.043	0.086	0.972	0.664
2	1	0.041	0.081	0.974	0.668
2	2	0.040	0.079	0.975	0.667
2	3	0.040	0.079	0.975	0.665

Table 2: Metrics as a function of number of inner PointFlow-refinement iterations (denoted l_i) and number of outer-loop scene-modeling passes (denoted l_o).

Model	Abs-rel	Abs-diff	$\delta < 1.25$	F-score
no 3d	0.067	0.134	0.952	0.551
single scale	0.041	0.080	0.973	0.662
avg feats	0.043	0.082	0.975	0.656
full	0.040	0.079	0.975	0.665

Table 3: Metrics for our ablation study. See Sec. 4.4 for descriptions of each condition.

bilizing and F-score decreasing slightly. Interestingly, the final iteration appears slightly detrimental.

Does Multi-Scale Scene Modeling Help? To test this, we (1) completely remove our multi-scale scene encoding from the PointFlow refinement, and (2) only use the coarsest scale $\mathbf{V}_3$, respectively denoted "no 3D" and "single scale" in Tab. 3. Without any scene-level information, our refinement breaks down, indicating the scene modeling is essential. The single scale model does slightly worse, confirming the effectiveness of our multi-scale encoding.

Do Multi-View Matching Features Help? We use a per-channel average instead of variance aggregation for each point in our feature-rich point cloud, denoted "avg feats" in Tab. 3. Most metrics, notably the F-score, suffer. This supports our hypothesis that multi-view matching is more beneficial for reconstruction than averaging.

For additional studies, see the supplementary material.

5. Conclusion

We present 3DVNet, which uses the advantages of both depth-based and volumetric MVS. We perform experiments with 3DVNet to show depth-based iterative refinement and multi-view matching combined with volumetric scene modeling greatly improves both depth-prediction *and* reconstruction metrics. We believe our 3D scene-modeling network bridges an important gap between depth prediction, image feature aggregation, and volumetric scene modeling and has applications far beyond depth-residual prediction. In future work, we will explore its use for segmentation, normal estimation, and direct TSDF prediction.

Acknowledgements: Support for this work was provided by ONR grants N00014-19-1-2553 and N00174-19-1-0024, as well as NSF grant 1911230.

References

[1] R. Q. Charles, H. Su, M. Kaichun, and L. J. Guibas. PointNet: Deep learning on point sets for 3D classification and segmentation. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 77–85, 2017. 3

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

[3] Julian Chibane, Thiemo Alldieck, and Gerard Pons-Moll. Implicit functions in feature space for 3D shape reconstruction and completion. In *Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, June 2020. 5

[4] Christopher Choy, JunYoung Gwak, and Silvio Savarese. 4D spatio-temporal ConvNets: Minkowski convolutional neural networks. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3075–3084, 2019. 6

[5] Angela Dai, Angel X. Chang, Manolis Savva, Maciej Halber, Thomas Funkhouser, and Matthias Nießner. ScanNet: Richly-annotated 3D reconstructions of indoor scenes. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. 1, 2, 6

[6] Arda Düzçeker, Silvano Galliani, Christoph Vogel, Pablo Speciale, Mihai Dusmanu, and Marc Pollefeys. DeepVideoMVS: Multi-view stereo on video with recurrent spatio-temporal fusion. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021. 1, 2, 5, 6

[7] WA Falcon and et al. PyTorch lightning. *GitHub. Note: https://github.com/PyTorchLightning/pytorch-lightning*, 3, 2019. 6

[8] Matthias Fey and Jan E. Lenssen. Fast graph representation learning with PyTorch Geometric. In *ICLR Workshop on Representation Learning on Graphs and Manifolds*, 2019. 6

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

[10] Ankur Handa, Thomas Whelan, John McDonald, and Andrew J. Davison. A benchmark for RGB-D visual odometry, 3D reconstruction and SLAM. In *International Conference on Robotics and Automation (ICRA)*, Hong Kong, China, May 2014. 2, 6

[11] Yuxin Hou, Juho Kannala, and Arno Solin. Multi-view stereo by temporal nonparametric fusion. In *International Conference on Computer Vision (ICCV)*, 2019. 1, 2, 6

[12] Sunghoon Im, Hae-Gon Jeon, Stephen Lin, and In So Kweon. DPSNet: End-to-end deep plane sweep stereo. In *International Conference on Learning Representations (ICLR)*, 2019. 1, 2

[13] Rasmus Jensen, Anders Dahl, George Vogiatzis, Engil Tola, and Henrik Aanæs. Large scale multi-view stereopsis evaluation. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 406–413. IEEE, 2014. 1, 6

[14] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations (ICLR)*, 2015. 6

[15] Arno Knapitsch, Jaesik Park, Qian-Yi Zhou, and Vladlen Koltun. Tanks and temples: Benchmarking large-scale scene reconstruction. *ACM Transactions on Graphics (TOG)*, 36(4), 2017. 1

[16] T. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie. Feature pyramid networks for object detection. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 936–944, 2017. 5

[17] 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)*, October 2019. 1, 2

[18] Zak Murez, Tarrence van As, James Bartolozzi, Ayan Sinha, Vijay Badrinarayanan, and Andrew Rabinovich. Atlas: End-to-end 3D scene reconstruction from posed images. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020. 1, 3, 6

[19] Songyou Peng, Michael Niemeyer, Lars Mescheder, Marc Pollefeys, and Andreas Geiger. Convolutional occupancy networks. In *European Conference on Computer Vision (ECCV)*, 2020. 5

[20] Olaf Ronneberger, Phillipp Fischer, and Thomas Brox. U-Net: Convolutional networks for biomedical image segmentation. In *Medical Image Computing and Computer-Assisted Intervention (MICCAI)*, volume 9351 of *LNCS*, pages 234–241. Springer, 2015. (available on arXiv:1505.04597 [cs.CV]). 2, 3

[21] 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

[22] Ayan Sinha, Zak Murez, James Bartolozzi, Vijay Badrinarayanan, and Andrew Rabinovich. DELTAS: Depth estimation by learning triangulation and densification of sparse points. In *European Conference on Computer Vision (ECCV)*, 2020. 1, 2

[23] Jürgen Sturm, Nikolas Engelhard, Felix Endres, Wolfram Burgard, and Daniel Cremers. A benchmark for the evaluation of RGB-D SLAM systems. In *International Conference on Intelligent Robot Systems (IROS)*, Oct. 2012. 2, 6

[24] Jiaming Sun, Yiming Xie, Linghao Chen, Xiaowei Zhou, and Hujun Bao. NeuralRecon: Real-time coherent 3D reconstruction from monocular video. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021. 1, 3, 6

[25] Mingxing Tan, Bo Chen, Ruoming Pang, Vijay Vasudevan, and Quoc V. Le. MnasNet: Platform-aware neural architecture search for mobile. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. 5

[26] K. Wang and S. Shen. MVDepthNet: real-time multiview depth estimation neural network. In *International Conference on 3D Vision (3DV)*, Sep. 2018. 1, 2

[27] Yue Wang, Yongbin Sun, Ziwei Liu, Sanjay E. Sarma, Michael M. Bronstein, and Justin M. Solomon. Dynamic graph CNN for learning on point clouds. *ACM Transactions on Graphics (TOG)*, 2019. 4

[28] Zhenpei Yang, Zhile Ren, Qi Shan, and Qixing Huang. MVS2D: Efficient multi-view stereo via attention-driven 2D convolutions. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021. 2

[29] 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)*, 2019. 1, 2, 3, 5

[30] 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

[31] Hongwei Yi, Zizhuang Wei, Mingyu Ding, Runze Zhang, Yisong Chen, Guoping Wang, and Yu-Wing Tai. Pyramid multi-view stereo net with self-adaptive view aggregation. In *European Conference on Computer Vision (ECCV)*, 2020. 1, 2

[32] 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, 5, 6

[33] Qian-Yi Zhou, Jaesik Park, and Vladlen Koltun. Open3D: A modern library for 3D data processing. *arXiv:1801.09847*, 2018. 6