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

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).

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

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.
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 regresses a TSDF in a coarse-
to-fine fashion using a 3D U-Net. NeuralRecon [24] im-
proves 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 vol-
ume using an RNN. The reconstructions these methods pro-
duce 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 $\{I_n\}$,
$n = 1, \ldots, N$ with corresponding known extrinsic and in-
trinsic camera parameters. Our goal is to predict $N$ depth
maps $\{D_n\}$ corresponding to the $N$ images. As a pre-
processing step, we define for every image $I_n$ 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 refer-
ence index to form the set $S_n = \{n, s_1, \ldots, s_M\}$.

Our pipeline is as follows. First, a small depth-prediction
network is used to independently predict initial coarse depth
maps $\{D_n^0\}$ for every frame $\{I_n\}$ using extracted image
features $\{F_n\}$ (Sec. 3.3). Second, we back-project our
$N$ initial depth maps to form a joint point cloud $X \subset \mathbb{R}^3$
(Sec 3.1). Because each point $p \in X$ is associated
with one depth map $D_n^0$ that has associated feature maps
$\{F_s : s \in S_n\}$, 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
$V_1, V_2, V_3$ (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 cur-
rent depth maps and step 4 run in the inner loop to refine
deep maps with the current scene model. We denote
the updated depth map after $l$ outer loop iterations of scene
modeling and $l_i$ inner loop iterations of updating as $D_n^{(l,l_i)}$.
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 pre-
dictions $\{D_n\}$ for every image $\{I_n\}$ (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 $\{D_n^0\}$ or subsequent up-
dated 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 inter-
mediate outputs at multiple resolutions. By nature of con-
struction, 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
$X \subset \mathbb{R}^3$ by back-projecting all depth pixels in all $N$ depth
maps. For our multi-view matching feature associated with
each point $p \in X$, 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
$p \in X$, given that $p$ belongs to depth map $D_n^0$, the equa-
tion for variance feature $\sigma^2(p)$, applied per-channel, is:

$$
\sigma^2(p) = \frac{1}{|S_n|} \sum_{s \in S_n} (F_s(\hat{p}_s) - \bar{F}_s(\hat{p}_s))^2
$$

(1)

where $\hat{p}_s$ is the projection of $p$ to feature map $F_s$. $\bar{F}_s(\hat{p}_s)$
is the bilinear interpolation of $F_s$ to point $\hat{p}_s$, and $\bar{F}_s(\hat{p}_s)$
is the average interpolated feature over all indices $s \in S_n$.
Intuitively, if $p$ lies on a surface it is more likely to have
low variance in most feature channels in $\sigma^2(p)$ 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
$C$ every $r = 8$ cm within the axis-aligned bounding box of
point cloud $X$ and define the voxel associated with each grid
point $c \in C$ as the $8 \times 8 \times 8$ cube with center $c$. We denote
the set of depth points that fall within a voxel with center $c \in C$
as $v(c) = \{p \in X : ||c - p||_\infty <= 8r\}$. We sparsify this grid by discarding $c \in C$ if no depth points lie within
the associated voxel, denoting this set of grid coordinates
as $\hat{C} = \{c \in C : v(c) \neq \emptyset\}$. For $c \in \hat{C}$, we produce a
feature for the associated voxel using PointNet [1] with max
pooling. The PointNet feature for each voxel is defined as:

$$
f_v(c) = \Delta h_\theta \left( \text{concat} [p - c, \sigma^2(p)] \right)
$$

(2)

where $h_\theta$ is a learnable multi-layer perceptron (MLP),
concat $[q, f]$ indicates concatenation of the 3D coordinates
$q$ with the feature channel of $f$ to form a feature with 3 ad-
ditional channels, and $\Delta$ is the channel-wise max pooling
operation. The result of this stage is a sparse feature vol-
ume $V_0$ with features given by Eq. 2 and coordinates $\hat{C}$.
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 $V_0$. From intermediate outputs of the U-Net, we extract three scales of feature volumes $V_1, V_2, V_3$ 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 $V_1, V_2, V_3$ 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
Next, we generate a variance feature \( \tilde{p} \) for each hypothesis point \( s \) and \( N \) priors in our offset prediction conditioned on all depth predictions using that scene model in the inner loop. We fix the input image size of 3DVNet to 320 \( \times \) 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 \( \times \) 64 and 160 \( \times \) 128 respectively. For every image \( I_n \), we denote these \( F^n_c \) and \( F^n_i \).

**MVSNet Prediction:** For the coarse depth prediction of image \( I_n \), we use a small MVSNet [29] using the reference and source coarse feature maps \( \{ F^n_s : s \in S_n \} \) to predict an initial coarse depth \( D^n_0 \). 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 \( F^n_r \). We fix our coarse depth map prediction size to 56 \( \times \) 56.

### Nested For-Loop Refinement

We denote the updated depth for each depth map as \( D^n_h \). We use initial depth predictions \( \{ D^n_0 \} \) and coarse feature maps \( \{ F^n_c \} \) to generate multi-scale scene encoding \( V_1, V_2, V_3 \). 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 \( \{ D^n_1 \} \). In early experiments, we found two iterations at 5 cm to be helpful. We re-generate our scene encoding using updated depths \( \{ D^n_1 \} \) and coarse feature maps \( \{ F^n_c \} \). 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 \( \{ D^n_2 \} \).

The updated depth for each depth map is the depth of point \( p + t \Delta d_p \) with respect to the camera associated with \( D^n_0 \).
We find our depth maps converge at this point.

Coarse-to-Fine Upsampling: In this stage, we upsample each refined depth prediction \( D_{3}^{(2,3)} \) to the size of image \( I_n \). 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 \( I_n \) 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 \( p \) in depth map \( D \), the smoothed depth \( \tilde{D} \) is a weighted sum of \( D \) in the \( 3 \times 3 \) neighborhood about \( p \):

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

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

4. Experiments

4.1. Implementation and Training Details


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

<table>
<thead>
<tr>
<th>( l_i )</th>
<th>( l_o )</th>
<th>Abs-rel</th>
<th>Abs-diff</th>
<th>( \delta &lt; 1.25 )</th>
<th>F-score</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0.070</td>
<td>0.137</td>
<td>0.949</td>
<td>0.559</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0.050</td>
<td>0.100</td>
<td>0.965</td>
<td>0.651</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>0.044</td>
<td>0.088</td>
<td>0.971</td>
<td>0.661</td>
</tr>
<tr>
<td>1</td>
<td>3</td>
<td>0.043</td>
<td>0.086</td>
<td>0.972</td>
<td>0.664</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>0.041</td>
<td>0.081</td>
<td>0.974</td>
<td>0.668</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>0.040</td>
<td>0.079</td>
<td>0.975</td>
<td>0.667</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>0.040</td>
<td>0.079</td>
<td>0.975</td>
<td>0.665</td>
</tr>
</tbody>
</table>

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 \( l_o \)).

<table>
<thead>
<tr>
<th>Model</th>
<th>Abs-rel</th>
<th>Abs-diff</th>
<th>( \delta &lt; 1.25 )</th>
<th>F-score</th>
</tr>
</thead>
<tbody>
<tr>
<td>no 3d</td>
<td>0.067</td>
<td>0.134</td>
<td>0.952</td>
<td>0.551</td>
</tr>
<tr>
<td>single scale</td>
<td>0.041</td>
<td>0.080</td>
<td>0.973</td>
<td>0.662</td>
</tr>
<tr>
<td>avg feats</td>
<td>0.043</td>
<td>0.082</td>
<td>0.975</td>
<td>0.656</td>
</tr>
<tr>
<td>full</td>
<td>0.040</td>
<td>0.079</td>
<td>0.975</td>
<td>0.665</td>
</tr>
</tbody>
</table>

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

Table: 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 \( V_3 \), 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


[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


