Using Synthetic Data Generation to Probe Multi-View Stereo Networks

Pranav Acharya* Daniel Lohn* Vivian Ross* Maya Ha Alexander Rich
Ehsan Sayyad Tobias Höllerer
University of California Santa Barbara, Santa Barbara, CA, USA
{pranavacharya, dlohn, vivianross, mayaha}@ucsb.edu, {anrich, holl}@cs.ucsb.edu,
ehsan@mat.ucsb.edu

Abstract

Synthetic data is highly useful for training machine learning systems performing image-based 3D reconstruction, as synthetic data has applications in both extending existing generalizable datasets and being tailored to train neural networks for specific learning tasks of interest. In this paper, we introduce and utilize a synthetic data generation suite capable of generating data given existing 3D scene models as input. Specifically, we use our tool to generate image sequences for use with Multi-View Stereo (MVS), moving a camera through the virtual space according to user-chosen camera parameters. We evaluate how the given camera parameters and type of 3D environment affect how applicable the generated image sequences are to the MVS task using five pre-trained neural networks on image sequences generated from three different 3D scene datasets. We obtain generated predictions for each combination of parameter value and input image sequence, using standard error metrics to analyze the differences in depth predictions on image sequences across 3D datasets, parameters, and networks. Among other results, we find that camera height and vertical camera viewing angle are the parameters that cause the most variation in depth prediction errors on these image sequences.

1. Introduction

Reconstruction from posed RGB images or depth scans is a growing field with applications ranging from autonomous driving to augmented reality. The task of 3D scene reconstruction involves processing RGB images and potentially depth scans and camera information to output a 3D model of the scene contained in the input images. The field has gained significant attention and opened up a multitude of opportunities for research, driving demand for data over the last decade.

In recent years, researchers have developed state-of-the-art machine learning models that perform 3D scene reconstruction [3, 6, 8, 11, 17, 18, 27]. A large amount of data is necessary to train these machine learning models, and several data generation platforms have already been created to satisfy this need for data [4, 19–22, 25, 26].

Our research is focused on determining how differences between such generated image sequences affect the performance of image-based 3D scene reconstruction networks trained on other data. Image-based 3D scene reconstruction encompasses a broad array of subtasks, some of which are typically performed deterministically. We evaluate neural networks that perform Multi-View Stereo (MVS), which involves using images of a scene from different viewpoints to construct a 3D model of that scene.

We introduce our own data generation platform to semi-automatically generate input data for these networks across different 3D scene datasets and camera parameter settings within the Unity real-time 3D development environment. The platform is currently focused on 3D indoor scene reconstruction, and to this end, allows researchers to load a variety of indoor scenes, artist-generated or scanned, and produce random but customizable ground-truth walk-through image sequences from them, to be used in 3D-computer-vision network training and testing.

We use our platform to generate image sequences for three carefully chosen 3D scenes, representing a range of realism from stylized synthetic (SUNCG [23]) to photorealistic (ArchViz [1]) and scanned (Matterport3D [5]). We analyze and gain useful insights into the performance of state-of-the-art 3D scene reconstruction networks, their architectures, and the properties of 3D scene datasets themselves. Furthermore, we offer recommendations on advantageous camera parameter settings and type of 3D scene datasets favoring accurate depth predictions.

2. Related Work

There are two methods through which training data for Multi-View Stereo algorithms is generally made available:

[*] Equal contribution

data generation frameworks, and ready-made datasets containing images, ground truth scans, camera poses, and other relevant information. Data generation frameworks generate synthetic training data on the fly, operating over either novel or existing 3D scene datasets. These 3D scene datasets can either consist of synthetic artist-created scenes or 3D reconstructions of real scenes. Our data generation platform currently only operates on indoor 3D datasets, focusing on 3D scene reconstruction networks. Therefore, indoor scene simulators are most applicable to our work. We first cover the inputs to data generation platforms, namely scanned and virtual 3D scene datasets, followed by a discussion of existing data generation platforms.

Image Datasets: Image datasets fall into two categories: those that are captured sequentially using an RGB-D video recording [7, 9, 15, 24] and those that are captured without specific ordering using placed camera rigs [5, 12, 13]. These datasets can be used without a simulator or the resulting 3D reconstruction can be used with a simulator.

We first cover datasets generated using precise camera rigs. Tanks and Temples [13] provides videos and ground-truth laser scans for both indoor and outdoor scenes. The DTU dataset [12] is captured using an industrial robot arm and a structured light sensor. It consists of "outside-in" views of objects rather than "inside-out." Matterport3D [5] is captured using a panoramic RGB-D camera which takes 360° images. Full 3D reconstructions with 3D segmentation are provided for 90 large-scale indoor scenes, often entire buildings. We selected the Matterport3D reconstructions for use with our tool for their size and completeness.

Generally, sequential datasets are of indoor scenes. TUM-RGBD [24], RGB-Dv2 [15], and 7Scenes [9] are smaller-scale datasets consisting of video sequences captured using a Microsoft Kinect or similar hardware. ScanNet [7] provides sequences for 1500 different scenes, with the 3D reconstruction, ground truth depth and both 2D and 3D segmentation provided. These datasets enable training of 3D reconstruction methods, but the ground truth reconstructions often have large holes. As such, we do not select these resulting reconstructed 3D scenes as input to our data generation tool.

Virtual Datasets: Virtual datasets consist of artist-created 3D scene models, generally in the form of a triangle mesh. A simulator is required to generate images from these 3D scene models, and we considered ICL-NUIM [10] and SUNCG [23] for use with our tool. ICL-NUIM consists of a small number of scenes with corresponding fly-through image sequences provided. SUNCG [23] consists of more than 45,000 indoor environments. We selected SUNCG for use with our tool.

Simulators: Embodied AI agents are robots trained to interact with a real-world environment in order to accomplish a given task. Indoor scene simulators are typically used as a benchmark for embodied AI agents [14,21,22,26], as training in these simulators is faster, safer, and less resource intensive than real-world training [21]. Indoor scene simulators are well-suited to generate training data for 3D reconstruction, as camera positions, RGB images, and depth maps are already outputted by many tools [19,21] for the purposes of training agents performing scene understanding tasks such as navigation [19].

Habitat [21] and MINOS [20] are platforms for embodied AI research with support for many 3D datasets, most notably SUNCG [23] and Matterport3D [5]. Matterport3D provides more realism compared to SUNCG, but SUNCG provides more customizability: its assets can be recombined to form new datasets [25]. The limitation of scanned 3D scene datasets such as Matterport3D is that each scene's lighting is baked into the scene's texture data as it is scanned. Hypersim [19] circumvents this limitation, providing researchers with the ability to enhance its own dataset of artist-created scenes by providing disentangled lighting data, allowing support for different lighting algorithms. Additionally, Hypersim releases the photorealistic rendered frames, camera trajectories and object segmentation data but the 3D models are not provided since they are from a commercial 3D asset package and should be purchased separately. The iGibson [22] and AI2-THOR [14] simulators add support for physics interactions between agents and objects in the scene, and multiple agents interacting in one scene, expanding the range of tasks embodied AI agents can be trained to perform.

Our tool is similar to existing scene simulators because we direct a virtual agent to navigate the scene and capture data during every movement. However, our tool is not used to evaluate scene navigation algorithms, as we delegate this task to an existing API. Additionally, our tool has the ability to load photorealistic datasets (see Fig. 1) on par with the datasets supported by iGibson, AI2-THOR, and Hypersim.

3. Methods

In this section, we describe our tool for generating image sequences given a 3D scene dataset as input.

We developed our tool using the Unity game engine. Unity provides essential real time 3D components such as shading, texturing, transformation and path finding. Our tool leverages these components to extract multi-channel 2D data from these 3D datasets by creating controlled fly-

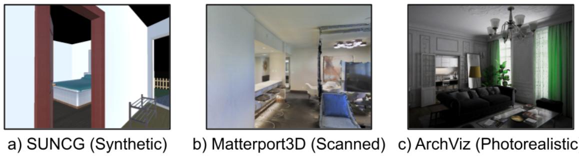
a) SUNCG (Synthetic) b) Matterport3D (Scanned) c) ArchViz (Photorealistic)

Figure 1. 3D Datasets used for evaluation.

Parameter	Description
Height	Camera distance above the floor
Camera Yaw	Clockwise camera offset from direction of travel
Camera Pitch	Camera angle below the plane at the camera's height and parallel to the floor
Sample Distance	Translational distance between image locations

Table 1. Parameters Used By Our Tool.

through sequences and rendering outputs such as RGB images, depth images and semantic segmentation masks.

We briefly describe the steps in our data generation pipeline (see Fig. 2). The input to our pipeline is a set of 3D scene models in the form of triangle meshes. These meshes can be textured or have material descriptions depending on the dataset being used. The output is a sequence of RGB images with corresponding camera extrinsic and intrinsic matrices, and depth maps. Optionally, we can render additional 2D information such as segmentation maps if the given 3D scene contains said information. The main steps to our pipeline are as follows. First, we load each scene into the Unity game engine. Second, we generate a walkable surface using the Unity NavMesh API. Third, we randomly generate a set of waypoints on the walkable surface. Fourth, we program a Unity camera to travel through a path in the scene using a random selection of waypoints. Along this path, we capture images, depth maps, and matrices at set translational intervals.

To study the effect of input data on MVS depth prediction networks, we add several user-specified camera parameters that affect how the image sequence is generated. Parameters are described in Table 1. The first three parameters (height, camera yaw, and camera pitch) modify camera viewing angle and height above the floor. The final parameter, sample distance, modifies how often images are captured.

4. Experiments

We generate image sequences of a selected subset of scenes from three different 3D scene datasets with differing values for each parameter in Table 2. When generating image sequences, the camera path in each house is kept constant. This is to ensure that all image sequences that are generated in a given house have identical paths, no matter the parameter settings. See Fig. 3 for a representation of different image sequences with a single parameter varied. With these image sequences, we test five pre-trained depth prediction networks.

Parameter	Values
Height (m)	0, 0.15, 0.3, 0.45, 0.6, 0.75, 0.9, 1.05, 1.2, 1.35, **1.5**, 1.65, 1.8, 1.95, 2.1
Camera Yaw (°)	-180, -135, -90, -45, **0**, 45, 90, 135, 180,
Camera Pitch (°)	-90, -60, -30, **0**, 30, 60, 90
Sample Distance (m)	**0.2**, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0, 1.1, 1.2, 1.3, 1.4, 1.5, 1.6, 1.7, 1.8, 1.9, 2.0

Table 2. Values for each Parameter. The bold values indicate the default parameter settings used to generate the control image sequences for each 3D dataset.

3D Scene Datasets We selected eight SUNCG [23] scenes, six Matterport3D [5] scenes, and one ArchViz [1] scene to load into Unity. These three datasets give us the widest range of scene data in that SUNCG consists of non-photorealistic synthetic scenes, Matterport3D consists of scanned data of real houses captured by camera rigs, and ArchViz consists of a photorealistic synthetic scene.

Parameters: See Table 2 for all parameter settings used during image sequence generation. When generating an image sequence in which we vary the settings for one parameter, we keep all the other parameter values at their default settings (indicated in bold in the table).

Depth Prediction Networks: To test how dataset parameters affect MVS algorithm performance, we use five existing neural networks with pre-trained weights to perform scene reconstruction on our data: DeepVideoMVS Pair Network (Pairnet), DeepVideoMVS Spacio-temporal Fusion (Fusionnet), Point-MVSNet (PMVS), Multi-view Stereo by Gaussian Process (GPMVS), and Fast-MVSNet (FMVS). Pairnet is a modified version of MVDepthNet that uses additional feature extraction, and Fusionnet extends Pairnet by including a ConvLSTM [8]. PMVS directly processes scenes as point clouds in a coarse-to-fine manner [6]. GPMVS uses an encoder-decoder model with a nonparametric Gaussian process that soft-constrains the bottleneck layer [11]. FMVS is a lightweight framework that uses sparse depth representation for fast and accurate MVS depth prediction [27].

Protocols: We evaluate the five networks above on generated image sequences for all 3D datasets and parameters. To make depth predictions on a given image, we use the previous and next image in the sequence as the source images to assist in MVS. We compare the predicted depth maps from each of the depth prediction methods to the ground-truth depth maps generated by our Unity tool and calculate the mean absolute relative depth error (abs-rel), given:

$$AbsRel = \frac{1}{n}\Sigma\frac{|d-d^*|}{d^*} \tag{1}$$

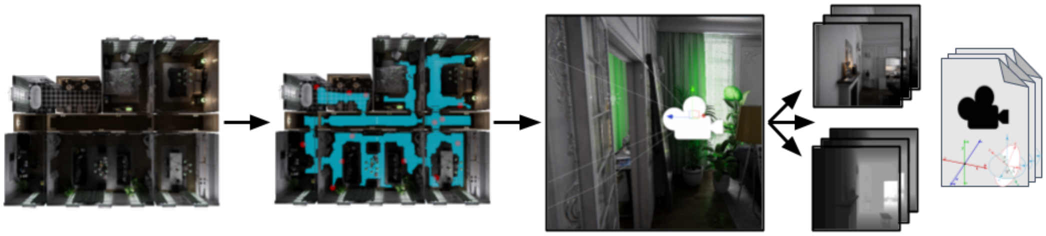

Figure 2. Our data generation pipeline. First, our tool generates a walkable surface and a set of randomized camera destinations. The camera moves along a path to each point, capturing data each frame the camera moves. More specific aspects of the camera's behaviour can be specified by the user as parameters. Our tool outputs RGB images, depth maps, and matrices that specify the camera's intrinsics and pose at each location.

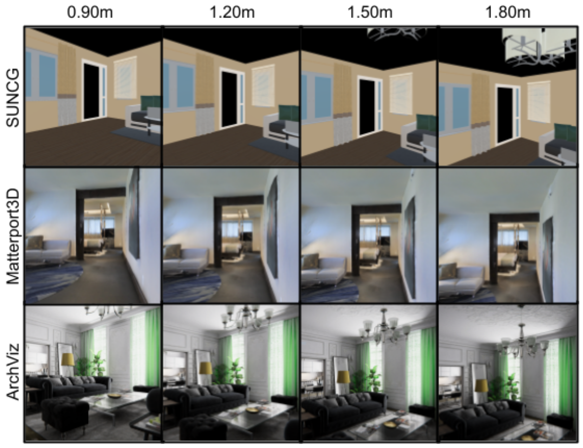

Figure 3. Sample captures made by our tool taken within each 3D scene dataset, with varying values for the camera height parameter.

where d and d^* are the predicted and ground-truth depth values respectively, and n is the number of pixels in the given image [8].

To verify that our testing script is correctly programmed to calculate the depth prediction error metrics for 2D image sequences, we run forward passes through each of the five networks with ScanNet. For Pairnet and Fusionnet, which were trained on ScanNet [8], the error metrics closely match the errors reported for these networks on ScanNet. For the other three networks, the error metrics obtained here are reasonable for methods not trained or finetuned on ScanNet. This verifies that our testing process did not produce any inaccurate or misleading error metrics for our following experiments on our generated image sequences. Table 3 holds these error metric results for our test script verification process:

Method	Abs-Rel Error
Pairnet	0.069
Fusionnet	0.061
PMVS	0.389
GPMVS	0.121
FMVS	0.274

Table 3. Test Set Error vs. Network.

In addition to this quantitative verification, we also use TSDF fusion to visually verify the depth prediction values produced by our test script as well as the intrinsic and extrinsic camera parameters.

With this error metric, we compare the accuracy of scene reconstruction on all of our image sequences across the five scene reconstruction networks, and the three 3D scene sets.

5. Results

Our experiments allow us to identify interesting trends by observing how changes in either the parameter values, network architectures, or the 3D datasets from which image sequences are generated affect depth-prediction accuracy. In subsection 5.1, we assess how the values for each parameter set affect the individual networks. In subsection 5.2, we report trends that applied to all networks. In subsection 5.3, we analyze how performance of the networks changes based on the 3D dataset from which the image sequences were generated. Finally, in subsection 5.4, we discuss how the network architectures of Pairnet, Fusionnet, and GPMVS might be causing them to perform very differently on image sequences with a low camera height parameter value.

5.1. Network Performance Optimization

We evaluate how each parameter affects the prediction accuracy of all evaluated networks.

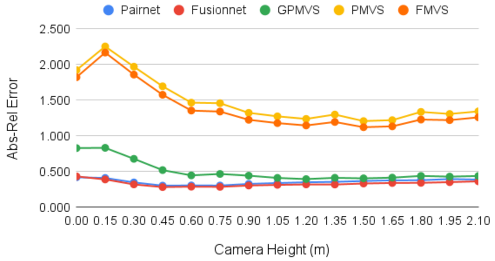

Figure 4. Average Error vs. Camera Height over all 3D datasets, split by network. Data is from Table 1 in the appendix.

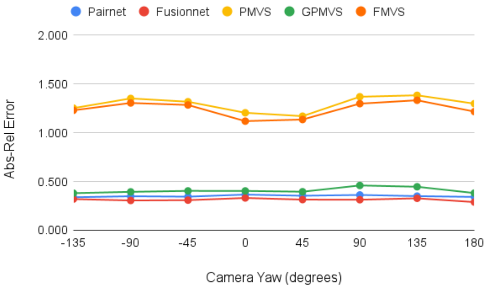

Figure 5. Average Error vs. Camera Yaw over all 3D datasets, split by network. Data is from Table 5 in the appendix.

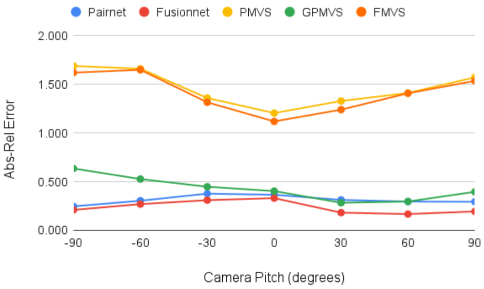

Figure 6. Average Error vs. Camera Pitch over all 3D datasets, split by network. Data is from Table 9 in the appendix.

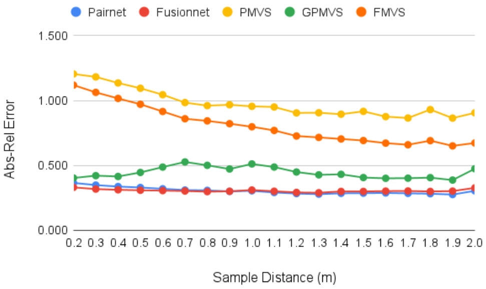

Figure 7. Average Error vs. Sample Distance over all 3D datasets, split by network. Data is from Table 13 in the appendix.

Height: For all the image sequences, the abs-rel errors are higher at camera heights right at ground level and at camera heights that are close to or at the ceiling. Pairnet and Fusionnet perform the best with camera heights of 0.3m to 0.75m based on Fig. 4, while the remaining networks perform the best on image sequences with camera heights of 1.05m to 1.65m.

Camera Yaw: The optimal yaw for each particular network varies for each 3D dataset. See Fig. 5 and Tables 6-8 in the appendix. For instance, while Pairnet and Fusionnet consistently have the least abs-rel error when tested on image sequences in which the camera is moving backwards relative to its viewing direction(yaw of ±90° to ±180°), the other three networks follow this pattern except on image sequences generated from Matterport3D where they perform the best with a +45° forward-facing yaw.

Parameters Networks	Camera Height	Camera Yaw	Camera Pitch	Sample Distance	Average
Pairnet	0.356	0.351	0.313	0.305	0.331
Fusionnet	0.331	0.314	0.239	0.306	0.297
GPMVS	0.502	0.408	0.421	0.446	0.444
PMVS	1.485	1.294	1.447	0.976	1.301
FMVS	1.386	1.241	1.399	0.809	1.208
Average	0.812	0.721	0.764	0.568	0.716

Table 4. Average Error for each parameter-network pair.

Parameters Networks	Height	Camera Yaw	Camera Pitch	Sample Distance
Pairnet	0.461	0.403	0.410	0.402
Fusionnet	0.494	0.363	0.360	0.406
GPMVS	1.075	0.517	1.199	0.841
PMVS	2.874	1.662	1.922	1.428
FMVS	2.879	1.335	1.858	1.378
Average	1.557	0.856	1.150	0.891

Table 5. Maximum Error Value for each parameter-network pair.

Parameters Networks	Height	Camera Yaw	Camera Pitch	Sample Distance	Average
Pairnet	0.044	0.018	0.064	0.028	0.038
Fusionnet	0.050	0.026	0.079	0.019	0.043
GPMVS	0.155	0.048	0.149	0.056	0.102
PMVS	0.326	0.105	0.218	0.115	0.191
FMVS	0.322	0.110	0.236	0.155	0.206
Average	0.179	0.061	0.149	0.074	0.116

Table 6. Standard Deviation of Error for each parameter-network pair.

Camera Pitch: The optimal pitch varies from network-to-network. See Fig. 6. Pairnet is the only network that has minimum error when tested on image sequences in which the camera is looking up. All the remaining networks have minimum error when the camera angle is either parallel to the plane of the floor or pointed downwards.

Sample Distance: We generally observe that the networks have their minimum abs-rel errors at sample distances of 1.2m to 1.9m, except for Fusionnet on Matterport3D in which the minimum error is at a sample distance of 0.8m. See Fig. 7 and Tables 14-16 in the appendix.

Network Variability: For all of the parameter sets, the data from Table 4 indicates that Pairnet and Fusionnet outperform the remaining three networks no matter which 3D dataset is used to generate the image sequences. For nearly all parameter sets, the networks in the ascending order of their average abs-rel error on the varied image sequences are Fusionnet, Pairnet, GPMVS, FMVS, and PMVS. Moreover, from Table 6, we see that Pairnet and Fusionnet also have substantially lower overall standard deviations across all parameter sets than the other three networks, meaning they produce relatively stable predictions and are less prone to unpredictable spikes in error. The key reason that PMVS and FMVS have significantly larger depth prediction errors than the other three networks could be due to the RGB-D datasets that the networks were trained on. Pairnet, Fusionnet, and GPMVS were trained on indoor scene RGB-D video sequences [8, 11], which are very similar to the image sequences that we generated from the three 3D datasets and tested the networks on. In contrast, PMVS and FMVS were trained on DTU, which includes RGB-D images of 3D models that are captured “outside-in” from surrounding cameras [6, 27]. Therefore, it is a strong possibility that not training on indoor scene image sequences resulted in PMVS and FMVS producing depth predictions on indoor scene image sequences that are significantly worse than depth predictions produced by the other three networks.

Conclusion: Through our experiments, we see that when generating image sequences to optimize the performance of existing 3D scene reconstruction networks, the best choices of the values for each camera parameter vary between networks. Additionally, based on their consistent out-performance of the other networks and low prediction variability, we hypothesize that Pairnet and Fusionnet are best suited to generalize to image sequences from different types of 3D datasets.

5.2. Trends Across All Networks

Table 5 shows the maximum abs-rel error value for each parameter-network pair across the three 3D datasets, followed by the average of those maximum values. Varying the height parameter yields the largest average maximum abs-rel error for most networks compared to other camera parameters. Table 6 shows that the average standard deviation of the abs-rel errors is the largest across all five networks when tested on image sequences that vary the height parameter. These two data points suggest that variations of the camera height have the most influence in determining the prediction error for all five networks. Moreover, Tables 5 and 6 show us that varying camera pitch yields the second-largest average maximum error and standard deviation of abs-rel errors. The fact that variations of camera height and pitch produce the two largest average maximum abs-rel errors and standard deviations of error leads us to hypothesize that all of the networks we use are affected most by image sequences with varying vertical camera views, since both camera height and pitch vary vertically in an “up and down” sense. In future work, we aim to address the high error and standard deviation caused by variations of camera height and pitch by including more image sequences for these parameters in the training set for these networks. Then, the networks can be tested to see if they produce higher accuracy depth predictions on these types of image sequence inputs after additional training.

5.3. Comparing 3D Dataset Results

Network Predictions based on 3D Datasets: We observe that image sequences generated from SUNCG generally result in larger depth-prediction errors for all networks compared to image sequences generated from Matterport3D or ArchViz. This trend holds true for all tested parameters, as shown by Figs. 8, 9, 10, and 11. This suggests a correlation between the realism of generated data and the accuracy of depth predictions made on this data. Specifically, because the networks were trained on scanned photorealistic datasets [6, 8, 11, 27], we hypothesize that as a result, the networks perform better on image sequences from scanned (Matterport3D) and qualitatively photorealistic (ArchViz) datasets as opposed to image sequences from SUNCG, which are non-photorealistic synthetic. See Fig. 1 for visual representations of these datasets. The data is inconclusive about whether it is Matterport3D or ArchViz whose image sequences cause the least depth prediction error among the five networks.

Similarities between 3D Datasets: In addition to comparing the depth prediction errors resulting from the image sequences generated by the three 3D scene datasets, we would like to judge how well models trained on each dataset will generalize to our desired application. In lieu of training networks on our data, we quantitatively evaluate how similar the datasets are to each other, with respect to the performance that randomly generated image sequences

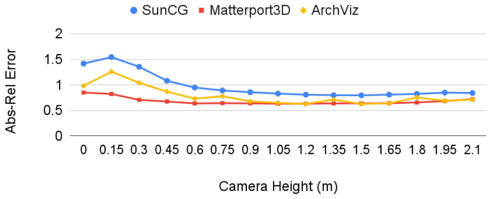

Figure 8. Average Error vs. Camera Height over all networks, split by 3D dataset. Data is from Tables 2-4 in the appendix.

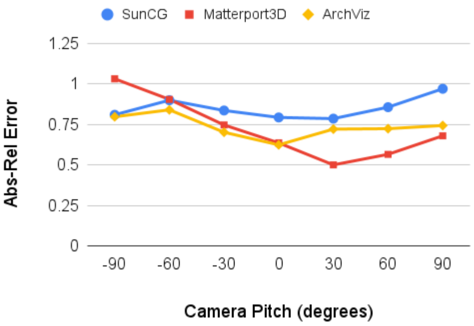

Figure 9. Average Error vs. Camera Pitch over all networks, split by 3D dataset. Data is from Tables 6-8 in the appendix.

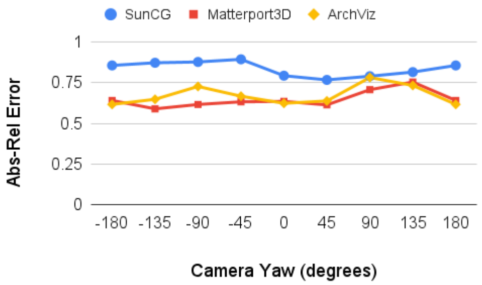

Figure 10. Average Error vs. Camera Yaw over all networks, split by 3D dataset. Data is from Tables 10-12 in the appendix.

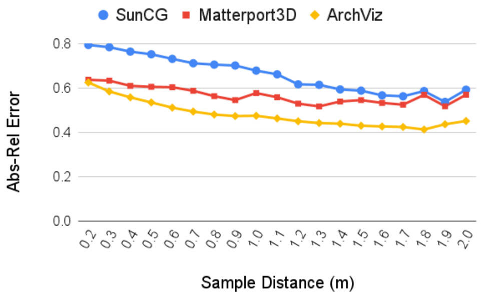

Figure 11. Average Error vs. Sample Distance over all networks, split by 3D dataset. Data from Tables 14-16 in the appendix.

from these datasets incur with the five pre-trained networks that we use in our evaluation. By calculating this similarity, we gain some understanding as to how a network trained on one dataset might perform on another dataset.

We analyze the pair-wise differences between our three datasets, namely Matterport3D to ArchViz, SUNCG to ArchViz, and SUNCG to Matterport3D. For each of our five pre-trained networks and each parameter in Table 2, we take the absolute value of the difference in error between the 3D datasets in the pair. We then average this difference across all networks and all parameters to get an average difference in error for each pair. This gives us an approximation of how similar each network's predictions are between datasets controlling for parameter. We use this as a proxy for how similar each given pair of datasets is. The results are detailed in Table 7. Finally, for each dataset, we sum the average difference in error between that dataset and each of the other two. The results are detailed in Table 8.

From Table 8, we see that SUNCG differs the most from the other two 3D datasets in abs-rel errors predicted from its generated image sequences, while ArchViz differs the least from the other two. From Table 7, we see Matterport3D and ArchViz are the closest pair. We hypothesize that the similarities of the textures between the scanned 3D dataset (Matterport3D) and the photorealistic synthetic scene (ArchViz) cause network predictions on image sequences derived from these datasets to be closer to each other than predictions between any of these datasets to the non-photorealistic synthetic scenes from SUNCG. Additionally, ArchViz has the least cumulative average difference, indicating that photorealistic synthetic scene datasets could be promising for generating training data.

3D Dataset Pair	Difference
Matterport3D to ArchViz	0.128
SUNCG to ArchViz	0.183
SUNCG to Matterport3D	0.233

Table 7. Pair-wise average absolute difference in error between datasets controlling for parameter and network.

3D Dataset	Difference
ArchViz	0.311
Matterport3D	0.361
SUNCG	0.416

Table 8. Cumulative average difference of 3D datasets. Obtained by summing two data values from Table 7 corresponding to each dataset.

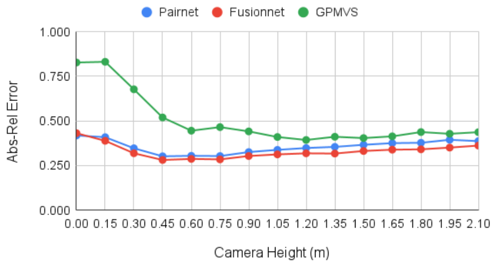

Figure 12. Graph of Average Error vs. Camera Height for Pairnet, Fusionnet, and GPMVS over all 3D Datasets. For camera height values less than 0.45m, GPMVS performs significantly worse than Pairnet or Fusionnet. Data is from Table 1 in the appendix.

5.4. Understanding Network Architectures

The manipulation of certain parameter settings allows our tool to generate data containing edge case situations in a higher quantity compared to other datasets that cannot be regenerated. The benefit of this edge case data is that differing performance trends may emerge across networks that were not trained specifically for this data. In this way, the performance of networks on the edge case data helps to elucidate how networks behave after training. For example, for small height values less than 0.45m on all datasets (Fig. 12), GPMVS degrades while Pairnet and Fusionnet perform well. In most other settings, the trends amongst these three networks are similar. For these parameter values, most of the generated images are dominated by the floor, which, as an example, does not contain enough unique features in the SUNCG dataset due to the absence of realistic lighting and use of tiled textures (see Fig. 13).

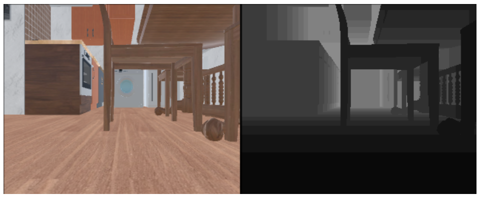

Figure 13. Example capture from SUNCG with Height = 0.15m. We believe the lack of unique features in the floor plane results in the breakdown of GPMVS predictions for low height values.

We briefly summarize a key difference amongst the three networks that might explain this data point. All three networks in question construct a cost volume on the reference camera frustum prior to depth prediction. In GPMVS, the cost volume is constructed using a difference of raw image RGB values from the reference and source images. In Pairnet and Fusionnet however, the cost volume is constructed using a dot product of deep features extracted from the reference and source images. After this cost volume construction, the architectures contain many similarities. All three use a U-Net-style 2D network to make a coarse-to-fine depth prediction from the cost volume. Fusionnet and GPMVS also modify deep features at the bottleneck layer of this UNet to improve depth predictions [8, 11]. Because of these similarities, we hypothesize that this difference in trends is caused by the absence of deep features in the cost volume construction of GPMVS.

Improving performance of depth prediction in untextured regions was a central motivation for the use of deep features in MVS. This data point supports this idea. More experimentation is required to explore this further; however, this is a promising avenue for future research with our tool.

6. Conclusion and Future Work

In this work, we utilize our data generation platform to generate 2D image sequences with user-specified camera parameter settings[1]. We then use these 2D image sequences to investigate both the performance of existing 3D scene reconstruction methods and the properties of the 3D scene datasets themselves. Our results show that Pairnet and Fusionnet, followed by GPMVS, produce significantly more accurate depth predictions across image sequences generated with all parameter settings from all 3D scene datasets. Matterport3D and ArchViz are more similar to each other than SUNCG is to either, which makes sense given the similarities of the textures of those two datasets. In addition, varying the camera height parameter causes the most variation in network accuracy for all of the networks across image sequences from all 3D scene datasets. Pairnet and Fusionnet use deep features instead of color density matching, which is a strong possible explanation for why they make better depth predictions at very low camera heights compared to GPMVS. One overarching takeaway from this is that user-specified values for a given camera parameter for generating the image sequence should be chosen based on the 3D scene reconstruction network that the generated image sequence will be used with.

By utilizing a widely used game engine as our realtime platform, we have the capability of easily extending the tool. Unity provides physically based shading, global illumination light mapping tools, and screen space effects such as ambient occlusion and screen space reflections, which are utilized in the ArchViz scene to create higher realism. Physically based path tracers such as Octane renderer [2] can be used in Unity for more photorealistic images.

One limitation of our project is that it is structured to be hosted on the Unity Editor and is difficult to export as a stand-alone application. This requires potential users to have a basic knowledge of Unity in order to use our tool.

None of our experiments produced state-of-the-art error metrics, so investigating whether this is due to the chosen parameter settings, or a lack of generalization, is an area for future work. We suspect that this is due to the networks being trained on different kinds of data. If the parameter settings are at fault, this leaves open the possibility of determining an optimal combination of values for all the camera parameters to achieve the lowest possible error. We could also pursue a transfer learning approach: performing further training of existing pre-trained methods using our generated image sequences to hopefully achieve signficant decreases in prediction error. Another path for further research is to investigate why prediction accuracy varies greatly between the five 3D scene reconstruction methods when tested on our data.

An interesting way to explore the best parameters for generating an image sequence in order to achieve best 3D reconstruction is to train an over-fitted scene representation network such as a NERF [16]. These networks in contrast to methods we explored, are not pre-trained on a dataset and are trained directly to a set of images and view directions to create a scene representation. These scene representation networks can be sampled as a signed distance function to create a 3D geometry of the model. This will give us the ability to directly compare the effect of our walk-through generation parameters on the quality of 3D reconstruction.

7. Acknowledgements

We thank Noah Stier for helpful discussions on 3D scene reconstruction networks. We also thank the mentoring team from the Early Research Scholars Program: Diba Mirza, William Eiers, and Aarti Jivrajani. Additionally, we thank Dr. Feng Yang from Google for his mentorship. This work was supported in part by NSF Grant #1821415.

[1] All code (excluding datasets) used is available at `https://github.com/vivianross06/Synthetic-Data-Generation-ICCV-2021`

References

[1] ArchVizPRO: The best way to learn real-time archviz visualization in unity. `https://oneirosvr.com/portfolio/archvizpro/`. Accessed: 2021-03-01.

[2] Octane for unity. `https://unity.otoy.com/`. Accessed: 2021-07-29.

[3] Aljaž Božič, Pablo Palafox, Justus Thies, Angela Dai, and Matthias Nießner. Transformerfusion: Monocular rgb scene reconstruction using transformers. *arXiv preprint arXiv:2107.02191*, 2021.

[4] Simon Brodeur, Ethan Perez, Ankesh Anand, Florian Golemo, Luca Celotti, Florian Strub, Jean Rouat, Hugo Larochelle, and Aaron Courville. Home: a household multimodal environment. *arXiv preprint arXiv:1711.11017*, 2017.

[5] Angel Chang, Angela Dai, Thomas Funkhouser, Maciej Halber, Matthias Niessner, Manolis Savva, Shuran Song, Andy Zeng, and Yinda Zhang. Matterport3D: Learning from RGB-D data in indoor environments. *International Conference on 3D Vision (3DV)*, 2017.

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

[7] Angela Dai, Angel X. Chang, Manolis Savva, Maciej Halber, Thomas Funkhouser, and Matthias Nießner. Scannet: Richly-annotated 3d reconstructions of indoor scenes, 2017.

[8] Arda Duzceker, Silvano Galliani, Christoph Vogel, Pablo Speciale, Mihai Dusmanu, and Marc Pollefeys. Deepvideomvs: Multi-view stereo on video with recurrent spatio-temporal fusion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 15324–15333, June 2021.

[9] Ben Glocker, Shahram Izadi, Jamie Shotton, and Antonio Criminisi. Real-time RGB-D camera relocalization. In *International Symposium on Mixed and Augmented Reality (ISMAR)*, 2013.

[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.

[11] Yuxin Hou, Juho Kannala, and Arno Solin. Multi-view stereo by temporal nonparametric fusion. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2019.

[12] 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.

[13] 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.

[14] Eric Kolve, Roozbeh Mottaghi, Winson Han, Eli VanderBilt, Luca Weihs, Alvaro Herrasti, Daniel Gordon, Yuke Zhu, Abhinav Gupta, and Ali Farhadi. AI2-THOR: An Interactive 3D Environment for Visual AI. *arXiv*, 2017.

[15] Kevin Lai, Liefeng Bo, and Dieter Fox. Unsupervised feature learning for 3D scene labeling. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2014.

[16] Ben Mildenhall, Pratul P Srinivasan, Matthew Tancik, Jonathan T Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. In *European conference on computer vision*, pages 405–421. Springer, 2020.

[17] 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 *ECCV*, 2020.

[18] Stefan Popov, Pablo Bauszat, and Vittorio Ferrari. Corenet: Coherent 3d scene reconstruction from a single rgb image. In Andrea Vedaldi, Horst Bischof, Thomas Brox, and Jan-Michael Frahm, editors, *Computer Vision – ECCV 2020*, pages 366–383, Cham, 2020. Springer International Publishing.

[19] Mike Roberts, Jason Ramapuram, Anurag Ranjan, Atulit Kumar, Miguel Angel Bautista, Nathan Paczan, Russ Webb, and Joshua M. Susskind. Hypersim: A photorealistic synthetic dataset for holistic indoor scene understanding.

[20] Manolis Savva, Angel X. Chang, Alexey Dosovitskiy, Thomas Funkhouser, and Vladlen Koltun. MINOS: Multimodal indoor simulator for navigation in complex environments. *arXiv:1712.03931*, 2017.

[21] Manolis Savva, Abhishek Kadian, Oleksandr Maksymets, Yili Zhao, Erik Wijmans, Bhavana Jain, Julian Straub, Jia Liu, Vladlen Koltun, Jitendra Malik, Devi Parikh, and Dhruv Batra. Habitat: A platform for embodied ai research. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2019.

[22] Bokui Shen*, Fei Xia*, Chengshu Li*, Roberto Martín-Martín*, Linxi Fan, Guanzhi Wang, Shyamal Buch, Claudia D'Arpino, Sanjana Srivastava, Lyne P Tchapmi, Kent Vainio, Li Fei-Fei, and Silvio Savarese. igibson, a simulation environment for interactive tasks in large realistic scenes. *arXiv preprint arXiv:2012.02924*, 2020.

[23] Shuran Song, Fisher Yu, Andy Zeng, Angel X Chang, Manolis Savva, and Thomas Funkhouser. Semantic scene completion from a single depth image. *Proceedings of 30th IEEE Conference on Computer Vision and Pattern Recognition*, 2017.

[24] 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.

[25] Yi Wu, Yuxin Wu, Georgia Gkioxari, and Yuandong Tian. Building generalizable agents with a realistic and rich 3d environment. *arXiv preprint arXiv:1801.02209*, 2018.

[26] Fei Xia, Amir R. Zamir, Zhi-Yang He, Alexander Sax, Jitendra Malik, and Silvio Savarese. Gibson Env: real-world perception for embodied agents. In *Computer Vision and Pattern Recognition (CVPR), 2018 IEEE Conference on*. IEEE, 2018.

[27] Zehao Yu and Shenghua Gao. Fast-mvsnet: Sparse-to-dense multi-view stereo with learned propagation and gauss-newton refinement. In *CVPR*, 2020.