OmniSyn: Synthesizing 360 Videos with Wide-baseline Panoramas

David Li[†,‡,*] Yinda Zhang[†] Christian Häne[†] Danhang Tang[†] Amitabh Varshney[‡] Ruofei Du[†,*]
[†] Google Research [‡] University of Maryland, College Park

Abstract— Immersive maps such as Google Street View and Bing Streetside provide true-to-life views with a massive collection of panoramas. However, these panoramas are only available at sparse intervals along the path they are taken, resulting in visual discontinuities during navigation. Prior art in view synthesis is usually built upon a set of perspective images, a pair of stereoscopic images, or a monocular image, but barely examines wide-baseline panoramas, which are widely adopted in commercial platforms to optimize bandwidth and storage usage. In this paper, we leverage the unique characteristics of wide-baseline panoramas and present OmniSyn, a novel pipeline for 360° view synthesis between wide-baseline panoramas. OmniSyn predicts omnidirectional depth maps using a spherical cost volume and a monocular skip connection, renders meshes in 360° images, and synthesizes intermediate views with a fusion network. We demonstrate the effectiveness of OmniSyn via comprehensive experimental results including comparison with the state-of-the-art methods on CARLA and Matterport datasets, ablation studies, and generalization studies on street views. We envision our work may inspire future research for this unheeded real-world task and eventually produce a smoother experience for navigating immersive maps.

Index Terms—360 image, virtual reality, view synthesis, panorama, neural rendering, depth map, mesh rendering, inpainting

1 Introduction

Recent advances in 360° cameras and virtual reality headsets have promoted the interests of tourists, renters, and photographers to capture or explore 360 images on commercial platforms such as Google Street View [1], Bing Streetside[1], and Matterport[2]. These platforms allow users to virtually walk through a city or preview a floorplan by interpolating between panoramas. However, the existing solutions lack the visual continuity from one view to the next and suffer from ghosting artifacts caused by warping with inaccurate geometry. While prior art reports successful view synthesis experiments in a set of perspective images [5, 8, 19, 22–24], a single image [29, 56], and a pair of stereoscopic panoramas with a narrow baseline [2], not much prior work addresses how we could synthesize an omnidirectional video with large movements, *i.e.*, using a *wide-baseline* pair of panoramas. Since wide-baseline panoramas are broadly adopted for capturing and streaming on commercial platforms, we envision view synthesis on this data can reduce the additional effort of converting to perspective images and leverage the full field-of-view for better alignment between the two panoramas.

Our goal is to synthesize 360° videos between wide-baseline panoramas and stream to consumer devices for an interactive and seamless experience (Fig. 1). Unlike past research which only synthesizes novel views within a limited volume [5, 29] or along a trajectory in rectilinear projection [19], our generated 360° video allows users to move forward/backward, stop at any point, and look around from any perspective. This unlocks a wide range of virtual reality applications such as cinematography [49], teleconferencing [51], and virtual tourism [14, 15].

Classical methods for view synthesis [22, 24] often rely on structure-from-motion [54] or multi-view stereo [21] pipelines to perform a sparse 3D reconstruction and develop algorithms to densify the reconstruction. Unfortunately, the existing approaches can hardly be applied to wide-baseline 360° images directly (Fig. 2). On the one hand, most existing works target perspective images which encounter *visual discontinuities* when objects move outside their field of view. On the other hand, applying monocular methods to multi-view scenarios leads to alignment issues between images as intermediate images are not fused from multiple views. Further, real-world street view images do not have a sufficiently dense layout to apply multiview stereo methods. So our research questions are: How can we achieve novel view synthesis between a pair of wide-baseline 360° images? How can we leverage the full field of view to align the pair of panoramas and inpaint occluded regions?

[*]Corresponding authors: dli7319@umd.edu and me@duruofei.com

[1]Bing Streetside: https://microsoft.com/en-us/maps/streetside
[2]Matterport: https://matterport.com

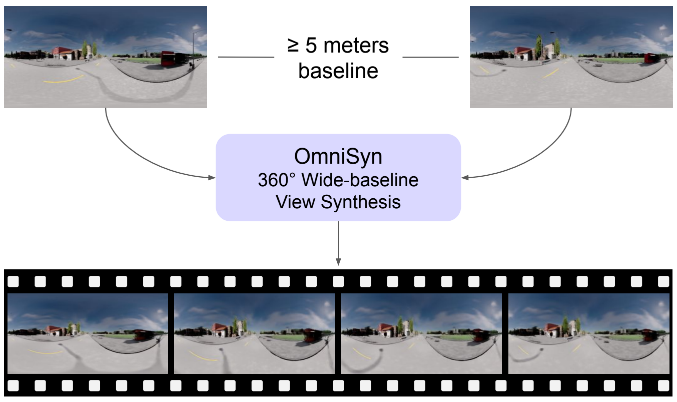


Fig. 1: Given two wide-baseline 360° images and poses, our goal is to synthesize a video sequence of intermediate frames with plausible movement and alignment between the input images.

To answer these questions, we contribute a new pipeline for 360° view synthesis using *wide-baseline* panoramas. Unlike the prior art, our inputs are a pair of 360° images which are at least 5 meters apart for street view scenes and 2 meters apart for indoor scenes. Our pipeline is comprised of a depth predictor, a 360° mesh renderer, and an image fusion network. We train our pipeline on datasets from CARLA and Matterport and compare it with the state-of-the-art monocular view synthesis pipeline for perspective images. Our contributions are as:

- Motivated by the goal of street view interpolation, we identify challenges associated with view synthesis between wide-baseline panoramas.
- We augment the classical view synthesis pipeline to address the challenges by using 360 cost volume for depth estimation, incorporating mesh rendering, and leveraging wide-baseline panoramas in the depth-estimation and fusion components.
- We conduct experiments against a modified version of SynSin, which is currently the state-of-the-art view synthesis pipeline. We also conduct ablation experiments to compare mesh rendering with point cloud rendering and identify the suitable scenarios for each.

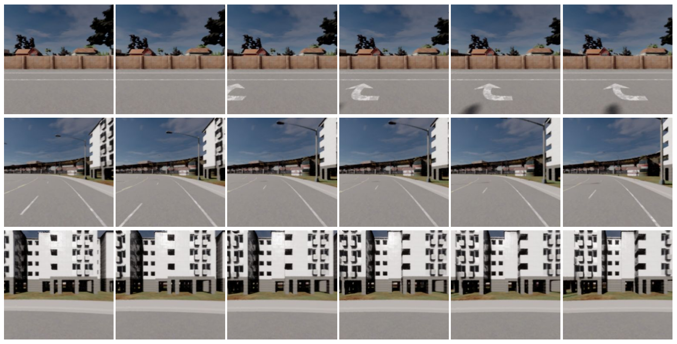


Fig. 2: Examples of perspective image sequences with 1 meter baselines. In perspective image pairs over long distances (e.g. 5 meters), large regions of each image are not visible from the other image. Therefore existing methods cannot be applied for wide-baseline 360° view synthesis by simply projecting to perspective images.

2 Related Work

Our work builds upon a rich literature of prior art on view synthesis and neural rendering. View synthesis has been studied in many setups such as dense image sequences, unstructured photo collections, as well as unique camera layouts. Here we provide a brief overview of the recent advances in perspective view synthesis and 360° view synthesis most relevant to our work.

2.1 Perspective View Synthesis

Existing work in view synthesis can be classified into one of the following categories: reconstruction-based, multiplane image (MPI) based, implicit-based, and warping-based methods.

Traditional methods in view synthesis reconstruct the underlying 3D geometry of the scene with multiview perspective images. Given a set of images with sufficient overlap, tools such as COLMAP [42,43] allow sparse 3D reconstruction using structure-from-motion and multi-view stereo techniques. Such 3D reconstructions can then be used to create meshes to render views from novel perspectives [20,24,58,59].

Later research focuses on view synthesis using layered representations including MPI [18,19,48,52,64], layered-depth images (LDI) [12,47,53], and multi-sphere images (MSI) [2]. Once the underlying representation is generated, these methods offer high-quality view-synthesis with standard mesh rendering. Layered methods work for both stereo inputs [19,38], as well as monocular methods [12,52].

Recent work on neural radiance fields (NeRF) [35,36,60] creates implicit representations for view synthesis from a set of perspective images. From these neural representations, views from new positions can be extracted using a volume rendering procedure. NeRF models can be trained using as few as 16 narrow-baseline images from a standard phone camera.

Warping and projection-based view synthesis are often used in low-data scenarios such as monocular and stereo view synthesis. Chen *et al.* [9] developed the transforming autoencoder which estimates the target depth and applies back-projection. Following this work, Wiles *et al.* [56] developed SynSin which performs forward-projection via differentiable point cloud rendering. By adding a feature encoder and GAN-based decoder, SynSin is even able to properly handle occlusions synthesizing views from a single image.

In wide-baseline view synthesis, researchers augment warping and projection with blending in developing view synthesis pipelines. Müller *et al.* [37] perform view synthesis by masking, projecting, and filtering background and foreground layers in a 3-stage process. Hobloss *et al.* [25] develop a hybrid approach for wide-baseline view synthesis with depth-based warping, two-step hole filling, and CNN-based blending procedures.

2.2 360° View Synthesis

Similar to view synthesis for perspective images, view synthesis for 360° images and videos has been studied with different goals and input scenarios. For instance, a 360° view synthesis loss can be used for self-supervised depth estimation [65].

In 360° view synthesis, researchers have developed pipelines for generating views from multiple cameras using traditional reconstruction pipelines. Hedman *et al.* [22] develop *Casual 3D photography*, an algorithm which creates 5-DoF 360° scenes from 180° fisheye cameras. With 50 input images along a ring, they can create a dense reconstruction which allows them to perform view synthesis with depth and normal map estimations. Similar approaches have also been applied to sets of 360° images [10,63] and 360° video [26].

Another traditional view synthesis technique involves light fields. Light fields involve capturing several images and considering each pixel as a result of a 4D plenoptic function evaluated from a light ray through free space. New images from different perspectives can then be generated by sampling from the existing set of light rays. Spherical light fields can be captured using a fisheye lens on a motorized setup [11].

360° view synthesis techniques have been used to create motion parallax for VR viewing [2–4,34,45]. Attal *et al.* develop MatryODShka [2] which uses multi-sphere images to add motion parallax to ODS video. Bertel *et al.* develop OmniPhotos [4], a method to perform 5-DoF view synthesis from a sequence of 360° photos along a roughly circular path with 1-meter diameter. Layered depth image (LDI) approaches have also been extended to spheres to offer real-time 6-DoF video playback [5,31].

Recently, there has also been a line of research on synthesizing street view images from satellite images. Lu *et al.* [33] synthesize street view images for urban areas by predicting the depth and semantics of satelite images. Shi *et al.* [46] develop and end-to-end pipeline for more suburban and rural areas by predicting multi-plane images (MPIs) from satellite imagery. Li *et al.* [30] synthesize street-view videos by predicting a latent point-cloud representation from satellite images. All techniques yield impressive results but do not closely resemble the true satellite images as satellite-to-street view is an extremely under-constrained problem.

Despite the significant efforts in view synthesis, existing techniques and pipelines are unsuitable for wide-baseline 360° view synthesis. Many often require dense sets of images or unique camera setups and promising monocular pipelines often fail to accurately align their synthesized views with ground-truth images. To address the challenge of wide-baseline 360° view synthesis from existing datasets, we have developed a pipeline tailored to this setup. Our pipeline is inspired by existing projection-based view synthesis pipelines but leverages 360° stereo inputs with varying baselines to estimate depth for accurate alignment between images. To enable inpainting and fusion over large baselines, we use mesh rendering which better represents the discontinuities of the underlying scene.

3 Method

OmniSyn, shown in Fig. 3, consists of three major components to synthesize novel views for wide-baseline panoramas: a stereo depth predictor, a differentiable 360° mesh renderer, and an image fusion network. OmniSyn takes two wide-baseline panoramas in equirectangular projection (ERP) and poses from the street view or Matterport's metadata as input.

Given two 360° ERP panoramas and their relative poses to a target position, our stereo depth predictor first estimates the depth of each panorama using a spherical sweep cost volume. Based on the estimated depths, we build a mesh representation for each panorama with discontinuities computed from depth estimates. Each mesh is rendered from the target position into a separate 360° panorama with a corresponding visibility map. Following this step, our fusion network joins the two panoramas together resolving ambiguities and inpaints any holes to produce the final 360° panorama.

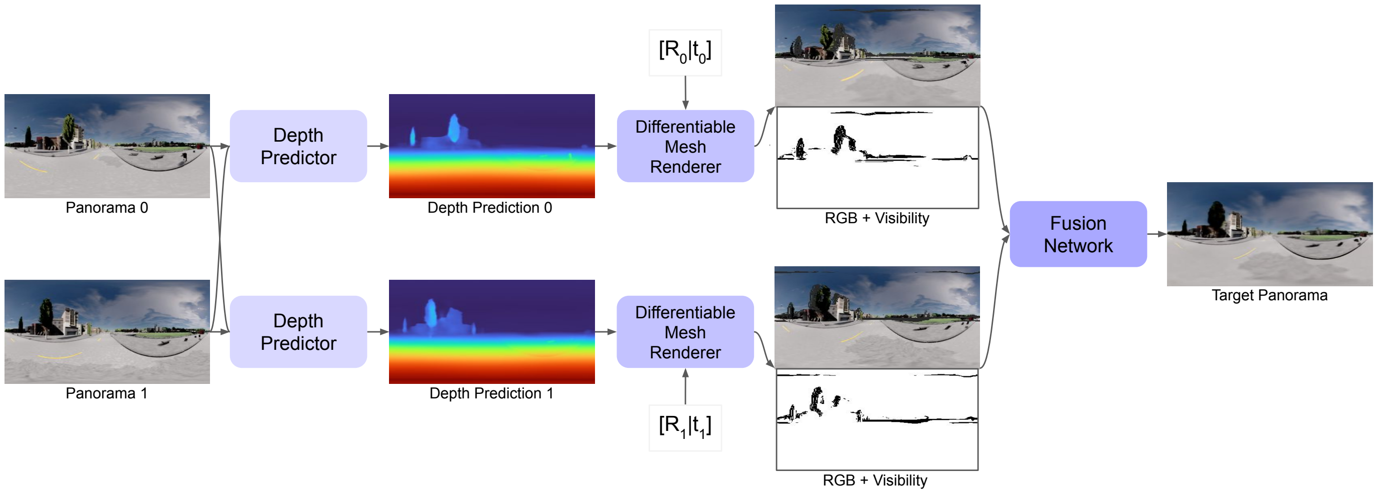


Fig. 3: Our 360° view synthesis pipeline consists of a stereo depth predictor, a 360° mesh renderer, and an image fusion network. All the three components are differentiable, while only the depth predictors and image fusion network have learnable parameters.

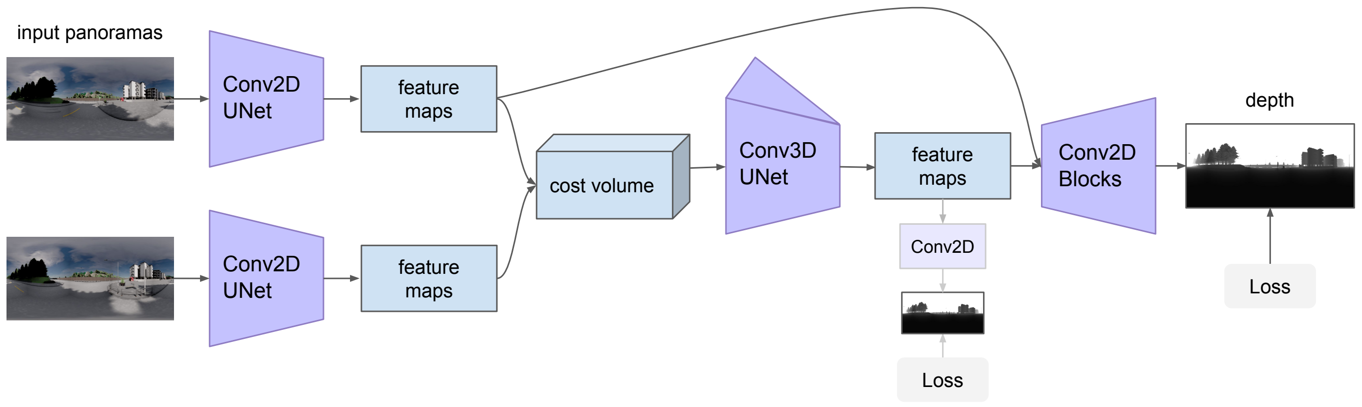


Fig. 4: An overview of our stereo depth prediction network for wide-baseline 360° panoramas. Our depth prediction network features a stereo path which estimates depth using a spherical cost volume and a monocular skip connection which allows semantic depth estimation for occluded regions.

3.1 Depth Prediction

To perform consistent depth prediction from wide-baseline panoramas, we build a network architecture inspired by StereoNet [28], modified for stereo 360° depth estimation. Our depth prediction network consists of three components: a 2D feature encoder, a 3D cost volume refinement network, and a 2D depth decoder. Stereo depth estimation allows the network to match features presented in both of the 360° images for aligned depth estimation while the monocular connection allows the network to predict depth for regions occluded in the other image. We follow StereoNet in first predicting a low-resolution depth. Next, instead of using the perspective cost volume, we leverage a spherical sweep [27, 57] cost volume of features. A refinement stage filters the cost volume and an upsampling stage guided by the feature map of the input image outputs the inverse depth prediction.

Our depth pipeline is shown in Fig. 4. We use a UNet [40] with 5 downsampling blocks and 3 upsampling blocks for the feature encoder, a 3D UNet with 3 downsampling and 3 upsampling blocks for the cost volume refinement network, and 2 convolutional blocks for the depth decoder. Similar to 360SD-Net [55], we input the vertical index as an additional channel for each convolutional layer in our depth prediction network, a technique also known as CoordConv [32]. This allows the convolutional layers to learn the distortion associated with the equirectangular projection (ERP). The final output convolutional layers in our depth network output one over the depth. To get the Euclidean depth, we take the inverse of this output.

3.2 Mesh Creation

To render each image from the novel viewpoint, we first create a spherical mesh for each input image. By using a mesh representation rather than a point cloud representation, we avoid density issues associated with creating point clouds from ERP images, as shown in Fig. 5. When moving large distances, point clouds created from ERP images contain widely varying levels of sparsity which are difficult to inpaint.

For a $W \times H$ resolution output image, we create a spherical mesh following the UV pattern with $2H$ height segments and $2W$ width segments. Next, vertices are offset to the correct radius based on the Euclidean depth d from the depth prediction stage. After creating the mesh and offsetting vertices to their correct depth, we calculate the gradient of the depth map along the θ and ϕ directions, yielding gradient images d_θ and d_ϕ. These gradient images represent an estimate of the normal of each surface. Large gradients in the depth image correspond to edges of buildings and other structures within the RGB

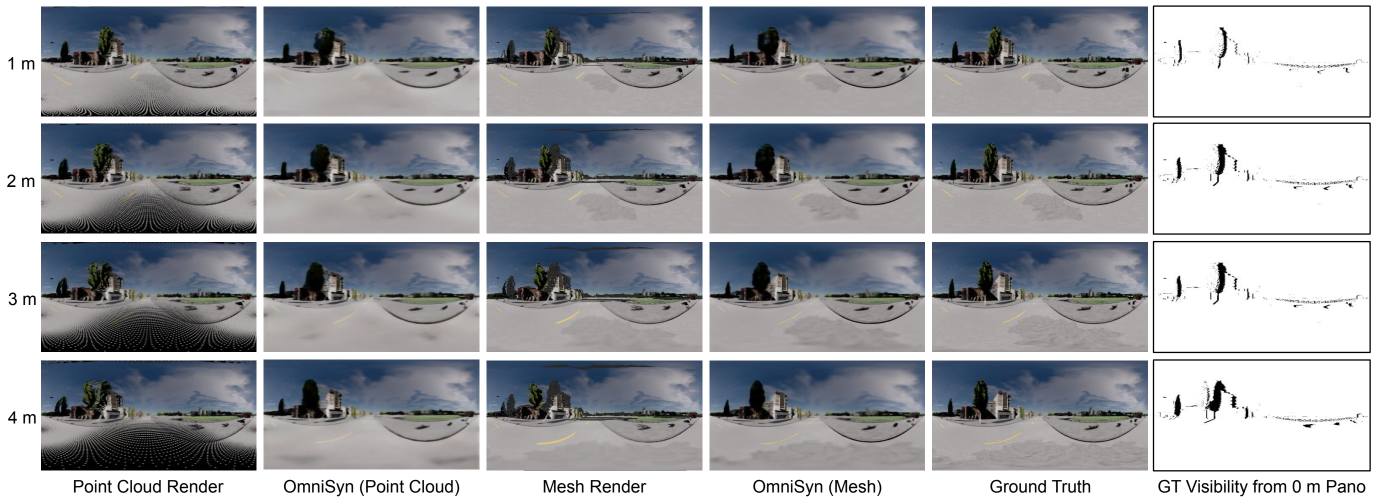


Fig. 5: A comparison between mesh rendering and point cloud rendering with ERP panoramas at $1-4$ meters of movement. As moving distance increases, the distribution of nearby points in the point cloud drastically shifts, leading to sparse regions in the point cloud render and lower overall quality. In contrast, our 360° mesh renderer more accurately projects the input with respect to the true visibility map.

Dataset	Method	IMAE ↓	IRMSE ↓	MAE ↓	RMSE ↓	1.05 ↑	1.10 ↑	1.25 ↑	1.25^2 ↑	1.25^3 ↑
CARLA	SynSin	0.0060	0.0090	1.19	5.34	0.8400	0.8837	0.9358	0.9799	0.9921
(ERP)	OmniSyn	**0.0042**	**0.0074**	1.22	6.20	**0.8473**	**0.8856**	0.9347	0.9770	0.9913
Matterport	SynSin	0.1320	0.1775	0.641	2.119	0.2021	0.3517	0.6187	0.8838	0.9667
(ERP)	OmniSyn	**0.0518**	**0.1142**	**0.282**	**0.741**	**0.6307**	**0.7706**	**0.8936**	**0.9573**	**0.9838**

Table 1: Quantitative results of Euclidean depth prediction on ERP images. Evaluating the accuracy of source depth prediction determines how well view synthesis results align between consecutive panoramas. For our stereo method, we present results using a 1-meter baseline between stereo images.

image. These surfaces have a normal vector perpendicular to the vector from the camera position. To classify discontinuities in the 3D structure, we identify areas where adjacent pixels are greater than some fixed value k apart. For these areas, we discard triangles within the spherical mesh to accurately represent the underlying discontinuity.

With the meshes created and discontinuities calculated, we use a modified version of PyTorch3D [39] to render the mesh from the new viewpoint to a 360° RGBD image. The mesh renderings contain holes due to occlusions in the original images. These holes are represented in the depth image as negative values, from which we extract a visibility mask, as shown in Fig. 3.

To adapt the built-in mesh renderer to output 360° images, we modify their rasterizer which project vertices from world-coordinates to camera-coordinates and finally to screen coordinates. Rather than multiplying vertex camera-coordinates by a projection matrix, we apply a Cartesian to spherical coordinates transformation and normalize the final coordinates to $[-1,1]$. However, doing this results in 2 issues: triangles wrapping around the left and right edges of the ERP images may be cut-off and straight lines in Cartesian coordinates may be incorrectly mapped to straight lines in ERP image coordinates. To address the first issue, we do 2 render passes, one rotated by 180°, and composite the passes together so that triangles which wrap around are not missing in the final render. We address the second issue by using a dense mesh to minimize the length of each triangle in the final image. One way to address the both issues simultaneously would be to render the 6 perspective sides of a cubemap and project the cubemap into an ERP image. However, this method incurs a significant performance and memory penalty from rendering 6 images.

3.3 Image Fusion

After rendering each mesh from the new viewpoint, holes appear in each rendering due to the occlusions in the synthesized view. A naive way to fill such holes is to alpha-blend both images based on their visibility maps. However, this may still leave holes in regions occluded in both images and lead to ghosting where objects are not perfectly aligned. Thus, we use a fusion network o fuse the two mesh renderings and inpaint the holes into a single consistent panorama. Specifically, we employ a 2D UNet. We first generate a binary visibility mask to identify holes in each rendered based on the negative regions in the mesh rendering depth image. Then we input both RGB mesh renderings and the corresponding binary masks into the fusion and inpainting network to get the final fused and inpainted RGB image.

Our fusion network is a 2D UNet with 6 downsampling blocks, 1 intermediate block, 6 upsampling blocks, and a final convolution layer. Each block consists of the following layers: Padding, Conv, LeakyReLU, Padding, Conv, and LeakyReLU. Downsampling is performed using average pooling at the end of each downsampling block. Upsampling is performed using bilinear interpolation at the beginning of each block. We use circular padding at each convolutional layer, simulating Circular CNNs [44], to join the left and right edges. The top and bottom of each feature map use zero padding. The inputs to our network are mesh renderings and visibility maps from each input panorama, totalling eight channels. The output is three channels of RGB representing the final inpainted frame.

3.4 Training

3.4.1 Loss Functions

Our network is end-to-end differentiable but training can also be performed in two supervised stages to increase modularity. In our experiments, we train each stage for 48 hours for a total of 96 hours of training on a single GPU. In the first stage, we train the stereo depth predictor. Our inputs for the depth predictor are two wide-baseline images in a sequence. The depth predictor includes two heads, an intermediate which predicts a low-resolution depth d_{pred_low} with n_{low} pixels based on only the cost volume and a final head which predicts a higher resolution depth d_{pred_hi} with n_{hi} pixels from the feature map and the cost volume. The intermediate head is used to ensure that gradients flow through the 3D cost volume UNet. Our loss function for depth is:

$$\ell_{depth} = \frac{1}{n_{hi}} \left\| \frac{1}{d_{gt}} - \frac{1}{d_{pred\_hi}} \right\|_1 + \frac{\lambda}{n_{low}} \left\| \frac{1}{d_{gt}} - \frac{1}{d_{pred\_low}} \right\|_1 \tag{1}$$

For our experiments, we use $\lambda = 0.5$.

In the second stage, we train the fusion network using sequences of 3 panoramas: (p_0, p_1, p_2). Mesh renders are generated from the first and last panoramas (p_0, p_2) using the pose of the intermediate panorama p_1. The fusion network receives these mesh renders and combines them to predict the full intermediate panorama p_{pred}. The ground-truth intermediate panorama p_1 is used for supervision. As the l_1 loss has been found to perform better for image-to-image applications [62], we use it to supervise the fusion network output:

$$\ell_{fusion} = \|p_1 - p_{pred}\|_1 \tag{2}$$

Our total loss is:

$$\ell_{total} = \ell_{depth} + \ell_{fusion} \tag{3}$$

3.4.2 Linearization of Arccos

Oftentimes in our pipeline, we need to convert between Cartesian and spherical coordinates for operations such as performing transformations or rasterizing to ERP images. The standard equations for converting from Cartesian to spherical coordinates are:

$$\theta = \text{arctan2}(z, x) \tag{4}$$

$$\phi = \arccos\left(\frac{y}{\sqrt{x^2+y^2+z^2}}\right) \tag{5}$$

$$r = \sqrt{x^2+y^2+z^2} \tag{6}$$

However, the derivative of $\arccos(\tilde{y})$ is $\frac{-1}{\sqrt{1-\tilde{y}^2}}$. Near the north and south poles, $\tilde{y}^2 = \frac{y^2}{x^2+y^2+z^2} \approx 1$, which leads to numerical issues computing the gradient in the backwards pass. To address this issue, we use a linear approximation for arccos for all points within α degrees of the north and south poles. This mitigates numerical issues by clipping the maximum and minimum gradient values near the poles, but preserving the sign of the gradient. In our experiments, we found that using $\alpha = 10$ degrees (≈ 0.174 rad) is sufficient for avoiding numerical issues in the backwards pass. During training, we use the following formula for ϕ:

$$\phi = \begin{cases} \arccos\left(\frac{y}{r}\right) & \text{if } \left|\frac{y}{r}\right| < \cos(\alpha) \\ \alpha \cdot \frac{1-y/r}{1-\cos(\alpha)} & \text{if } y > 0 \text{ and } \left|\frac{y}{r}\right| \geq \cos(\alpha) \\ \pi - \alpha \cdot \frac{1+y/r}{1-\cos(\alpha)} & \text{if } y \leq 0 \text{ and } \left|\frac{y}{r}\right| \geq \cos(\alpha) \end{cases} \tag{7}$$

4 Experiments

We conduct experiments comparing our pipeline to the state-of-the-art view synthesis pipeline, SynSin [56]. Our experiments are conducted on synthetic outdoor scenes from CARLA [13] and real-world indoor scenes from Matterport3D [7]. To evaluate how models from each pipeline adapt to variable wide-baseline scenarios, our evaluations focus on view-synthesis results across variable distances and source depth accuracy which is essential for alignment between images synthesized from consecutive images.

4.1 Datasets and Metrics

We conduct experiments on synthetic street scenes from CARLA [13] as well as real-world indoor scenes from Matterport3D [7]. For each dataset of scenes, we compare source depth prediction and view synthesis results. Source depth accuracy provides insight into how well view synthesis results align with adjacent panoramas. View synthesis results provide insight into the accuracy of inpainting and fusion.

For CARLA, we use scenes from Town 1 to Town 6. Towns 1, 2, 3, and 4 are used as training towns with Town 5 used for validation. Town 6 is used for testing. We customize each scene to normalize weather conditions and to remove all cars and pedestrians. For each town, we generate sequences of 1000 frames from each of the first 40 starting points. Each frame consists of an RGB panorama, depth panorama, and pose metadata. Panoramas are created by rendering the 6 sides of cubemaps at 256×256 for each frame and stitching them into 1024×512 ERP images. During training, we resize and subsample each sequence of frames to create 256×256 image sequences with a fixed distance between frames. We use baselines from 1 meter to 6 meters with 1 meter intervals during training. For illustrative purposes, we display results with a 2 : 1 aspect ratio by resizing the 256×256 outputs.

For Matterport3D, we use Habitat-Sim [41] to generate frames at training and inference time. Similar to CARLA, we generate 6 256×256 perspective images representing cubemap sides and stitch them into a single 256×256 ERP image. Our train and test split follows that of SynSin. During training and evaluation, we use sequences of 3 panoramas with a fixed baseline from 0.25 meters to 1 meter with 0.25 meter intervals. The middle panorama is synthesized based on the first and last panorama in the sequence.

For comparing the accuracy of Euclidean source depth prediction, we use mean absolute error of the inverse depth (IMAE) and root mean squared error of the inverse depth (IRMSE) following the KITTI Depth Completion Evaluation benchmark[3]. Inverse depth weights errors in closer depths greater than errors in further depths and is directly proportional to disparity in perspective images. We also present the standard mean absolute error (MAE) and root mean squared error (RMSE) results. Inverse depth metrics are measured in one over meters and standard depth metrics are measured in meters. Following existing depth-estimation papers [61], we present $\delta < [1.05, 1.10, 1.25, 1.25^2, 1.25^3]$. To omit outliers and missing regions in each dataset, such as the sky, we compute results with respect to valid depth regions in each dataset between 1 and 50 meters. Our full depth prediction results are shown in Table 1. Qualitative results for depth prediction are shown in Fig. 6 using the Turbo colormap[4].

To compare view synthesis results for spherical ERP images, we use Weighted-to-Spherically-uniform PSNR (WS-PSNR) [50]. View synthesis results are computed for a variety of distances for both datasets. We also provide results of qualitative comparison in Fig. 8 and Fig. 9 on both CARLA and Matterport3D datasets.

4.2 Comparison to SOTA

We compare the depth and view synthesis results of our method to that of the current state-of-the-art view synthesis method, SynSin [56], which also uses geometric based warping. To adapt SynSin which is designed for perspective images to 360° images, we modify their perspective point cloud renderer to project 360° point clouds and render 360° ERP images.

While SynSin is designed to work with only one image, we also compare with a 2-view version of SynSin in Fig. 7. For the 2-view version of SynSin, the depth predictor and feature encoder operates independently for each input image. Then both latent point clouds are combined by concatenating the list of points. A single output image is generated by decoding a single feature map from the combined point cloud.

For both SynSin and OmniSyn, we train in a supervised manner using ground-truth depth from both datasets. While SynSin provides compelling results with small movements, OmniSyn performs better in aligning with the second ground truth image and inpainting occluded areas with large movements.

[3]KITTI benchmark: `http://cvlibs.net/datasets/kitti/eval_depth.php?benchmark=depth_completion`

[4]Turbo colormap: `https://ai.googleblog.com/2019/08/turbo-improved-rainbow-colormap-for.html`

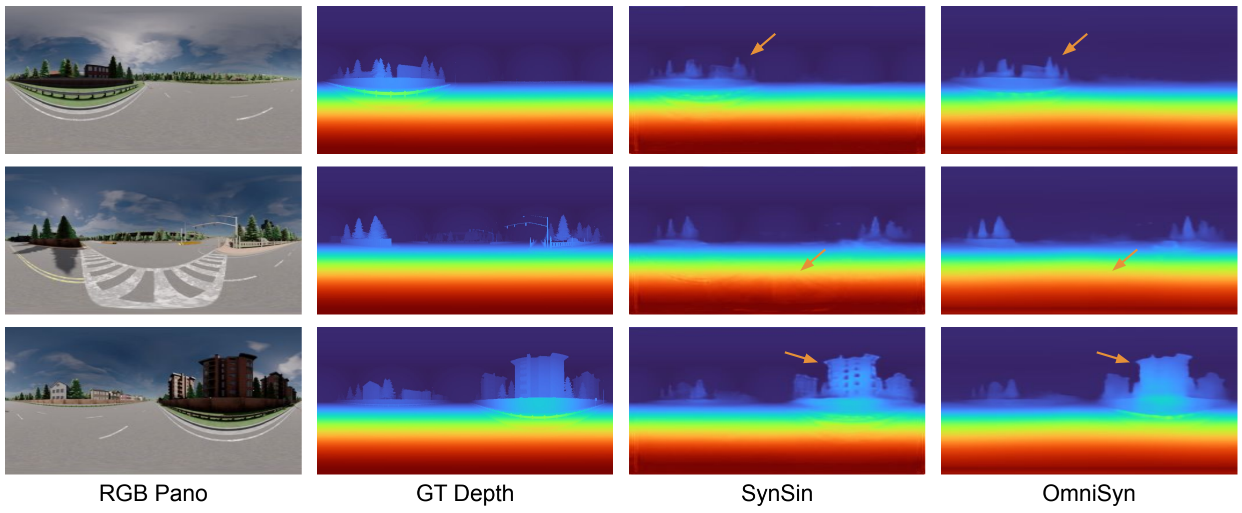
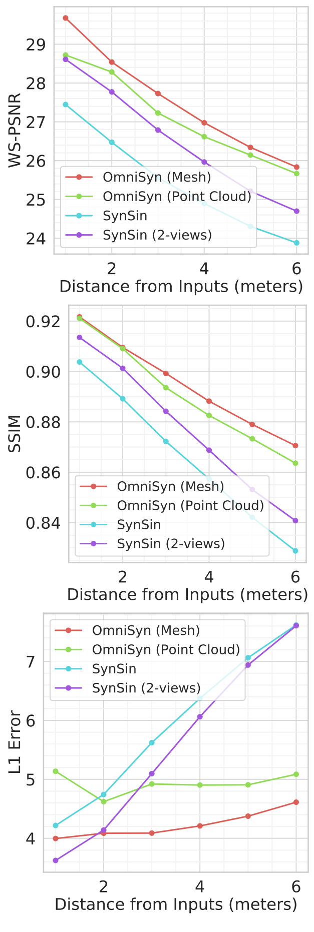
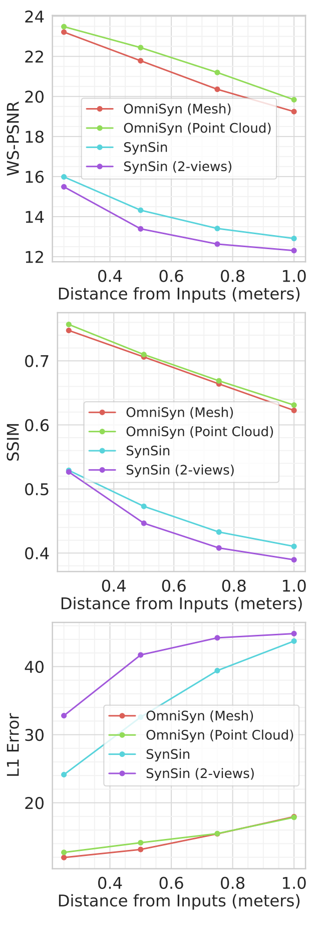


Fig. 6: Qualitative results for Euclidean depth prediction on the CARLA dataset.



(a) Carla Synthesis Quality



(b) Matterport Synthesis Quality

Fig. 7: Quantitative view synthesis results of OmniSyn on Carla and Matterport3D datasets. For 360° view synthesis, OmniSyn outperforms SynSin for both indoor and outdoor scenes. However, there is a quality trade-off associated with using mesh over point clouds in indoor scenes.

4.3 Robustness to Wide Baseline

To determine how different 3D representations respond to varying baselines of inputs, we conduct an ablation study comparing the results of point-cloud rendering and mesh rendering. As shown in Fig. 5, point cloud representations benefit from small movements as discontinuities do not need to be explicitly calculated based on normal estimates. However, for large movements needed in wide-baseline view synthesis, point clouds suffer from sparsity in their representation, leading to worse inpainting results than the mesh renderer. Quantitative results are shown in Fig. 7.

4.4 Generalization to Real Street View Images

To qualitatively evaluate how well our model generalizes to real street view images, we run our stereo model over selected images from Google Street View [1]. We select suburban scenes without moving cars or pedestrians to match the static scene scenario of the CARLA training dataset. For each set of input images, we generate two intermediate images using an estimated pose from geographic (latitude, longitude, heading) metadata. Qualitative results are shown in Fig. 10.

5 Discussion

We have shown that OmniSyn outperforms the state-of-the-art monocular view synthesis pipeline on CARLA and Matterport datasets. While our pipeline requires a pair of wide-baseline panoramas, such data is widely available on commercial platforms such as Google Street View and Bing Streetside. We discuss our key findings and limitations.

5.1 Observations

OmniSyn handles occlusions using stereo inputs and a fusion network that fuses stereo inputs and inpaints occluded regions in both images whereas SynSin handles occlusions using latent features and a GAN-based decoder network. When comparing quantitative results from SynSin and OmniSyn, we see that stereo inputs outperform GAN-based feature decoding. This is especially true for indoor scenes as shown in Fig. 9.

In the case of indoor scenes from Matterport3D, monocular depth prediction yields less accurate depths than stereo depth prediction as shown in Table 1 due to the greater variability of scenes. This makes monocular methods unsuitable for wide-baseline view synthesis from stereo panoramas where accurate depth is crucial for aligned view synthesis. We see in Fig. 7b, the 2-view version of SynSin actually performs slightly worse on the indoor scenes due to the inaccurate monocular depth. However, for outdoor scenes from Carla where the depth of the road is fairly consistent between all images, monocular depth prediction yields similar results to our stereo depth. This allows OmniSyn and the 2-view version of SynSin to both outperform the single-view SynSin.

Comparing mesh rendering and point cloud renderings, we see in Fig. 5 that point cloud rendering has high sparsity in regions closer to

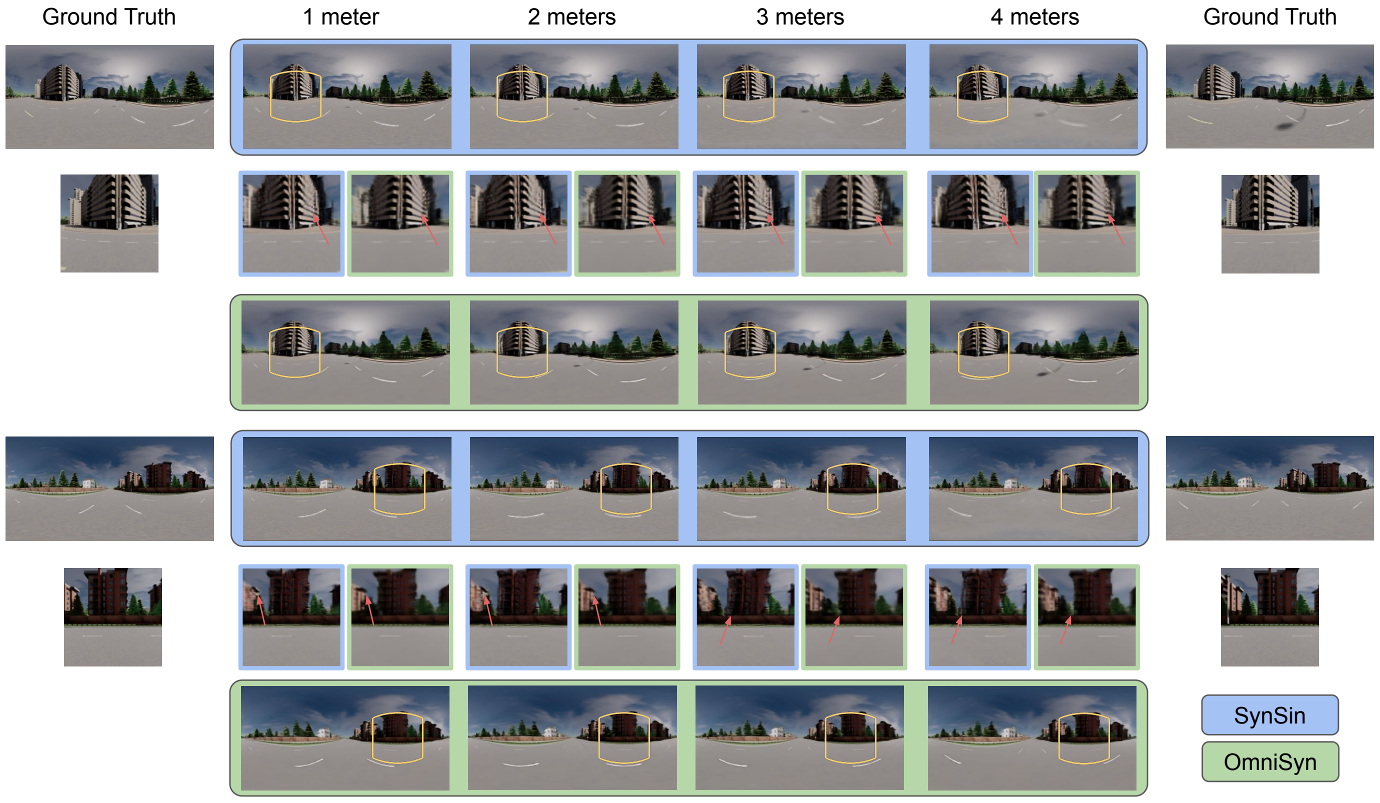


Fig. 8: Qualitative results of 360° view synthesis on outdoor scenes of the CARLA dataset with inputs 5 meters apart. By leveraging stereo panoramas for depth prediction and synthesis, our method is able to generate views to accurate metric scale distance and render sides of buildings that may be unclear or less sharp when generated from only a single view.

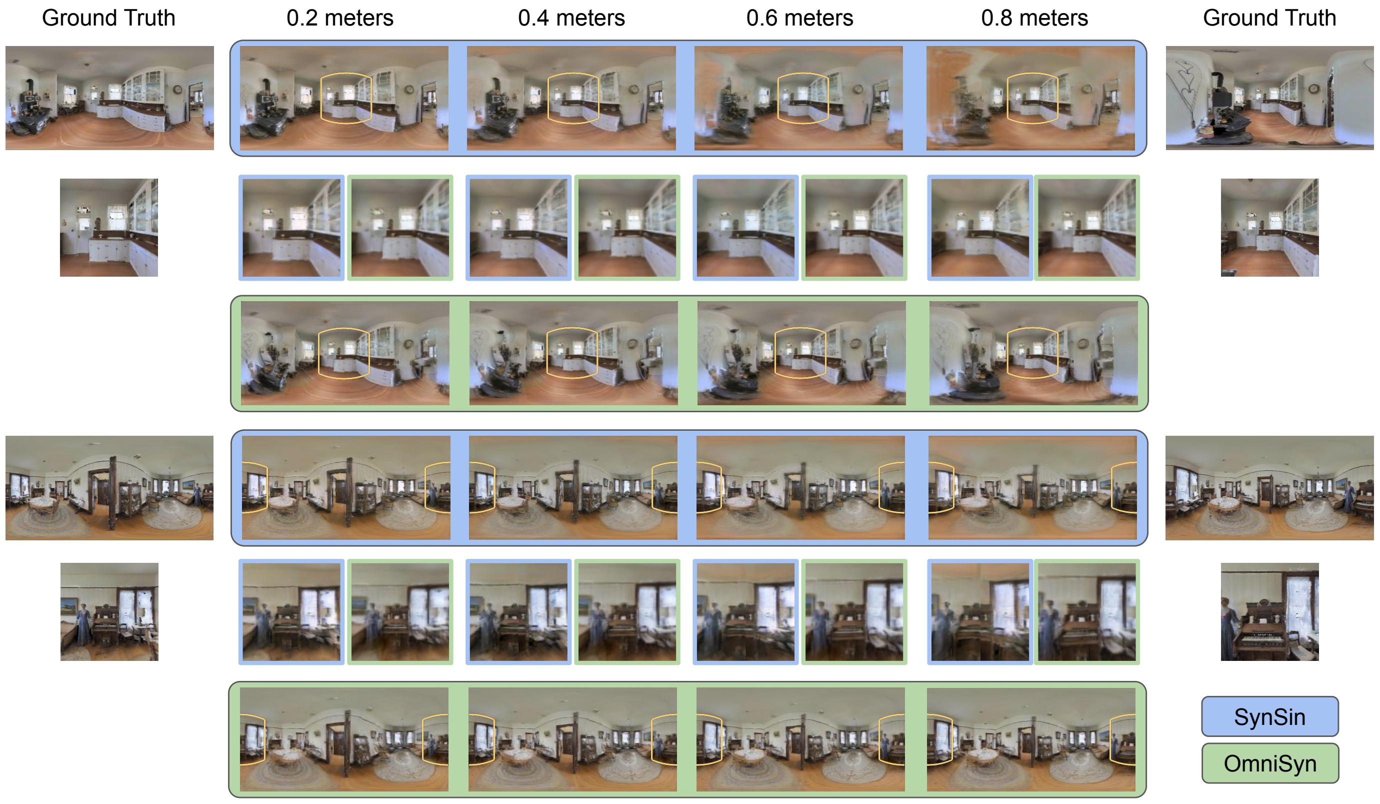


Fig. 9: Qualitative results of 360° view synthesis on indoor scenes of the Matterport3D dataset. The pair of ground truth images are spaced 1 meter apart.

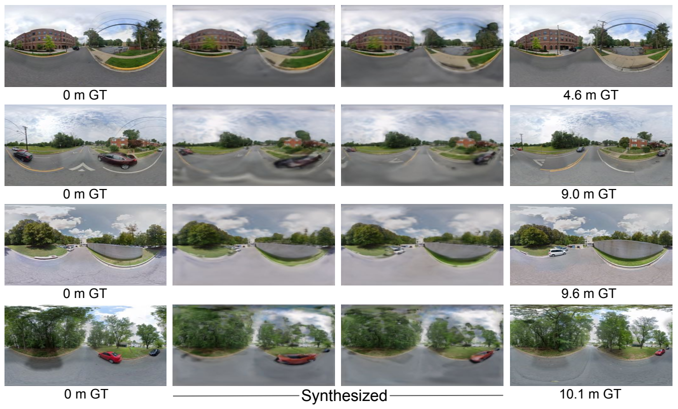


Fig. 10: Generalization of our model trained on the CARLA dataset to suburban street view images.

the camera, forcing the inpainting or fusion networks to fill-in more of the road. In scenarios where there are distinct textures on the road, such as lane markers and traffic symbols, this causes OmniSyn to generate incorrect textures. When looking at quantitative results, OmniSyn with point cloud rendering achieves worse WS-PSNR results than with mesh rendering. Therefore for outdoor scenes where the depth to the closest regions is small relative to the amount of movement, we suggest using mesh rendering over point cloud rendering to avoid such sparsity issues.

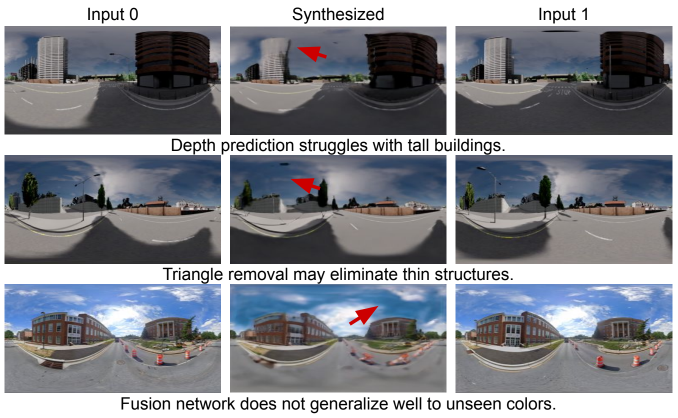

Input 0
Synthesized
Input 1
Depth prediction struggles with tall buildings.
Triangle removal may eliminate thin structures.
Fusion network does not generalize well to unseen colors.

Fig. 11: Three types of failure cases which may cause artifacts.

5.2 Limitations

Running our CARLA-trained OmniSyn model on Google Street View images, as shown in Fig. 10, we see that the outputs are reasonable for objects within the CARLA dataset such as buildings and roads. However, thin objects such as wires and tree branches are not well represented in the output. Furthermore, objects such as cars that are not included in the CARLA training procedure are poorly rendered. Similar to the prior art in view synthesis, our pipeline currently focuses on static scenes. While this assumption may hold in street view images of sparsely-populated suburban areas, most real street view images are taken in a dynamic environment filled with moving cars, pedestrians, and objects. One way to address this would involve extending object motion detection methods [6] to sparse 360° street view imagery. High-quality view synthesis for real 360° street view scenes continues to remain an open challenge.

We identify three types of failure cases in Fig. 11. First, spherical sweeping struggles to estimate depth for tall buildings due to distortion in the ERP projection and sweeping levels being more concentrated on closer depths. Second, triangle culling may lead to thin objects being removed. Third, the fusion network does not generalize well to unseen objects and colors. One may overcome these limitations by training with a diverse synthetic data which includes moving objects or with large-scale street view datasets. The depth predictor may also be further augmented with rectangular filters [66] and a quaternion loss function [17].

Our current pipeline and primary results are ran on 256×256 resolution images which is unsuitable for high-resolution VR and AR experiences as users only see a limited portion of 360° images at any given moment. One of the benefits of using a mesh representation, or similar MPI and LDI representations, is that meshes can be textured with high-resolution images, regardless of the vertex density of the mesh. Therefore, higher resolution results can be achieved by combining a low-resolution depth prediction with a high-resolution mesh rendering as done in previous work [2].

6 Conclusion

In this paper, we examine the task of intermediate view synthesis for wide-baseline 360° panoramas, typically ≥ 5 meters apart. We start by evaluating whether state-of-the-art view synthesis techniques are suitable for creating 360° views. From our experiments, we observe the following: First, current monocular methods hallucinate content that may be inconsistent and unsuitable for wide-baseline 360° view synthesis. Using stereo images allows the network to more accurately synthesize views and estimate depth. Second, for 360° view synthesis, point cloud renderings incur unnecessary sparsity in nearby objects such as roads. Using mesh rendering better represents the underlying visibility for 360° images. Based on these observations, we develop OmniSyn which leverages 360° stereo depth estimation, mesh rendering, and 360° fusion to synthesize plausible 360° street view panoramas from static scenes. We envision this line of research may give rise to a wide range of virtual reality applications with depth-based real-time interaction [16].

References

[1] D. Anguelov, C. Dulong, D. Filip, C. Frueh, S. Lafon, R. Lyon, A. Ogale, L. Vincent, and J. Weaver. Google street view: Capturing the world at street level. *Computer*, 43(6):32–38, 2010. doi: 10.1109/MC.2010.170

[2] B. Attal, S. Ling, A. Gokaslan, C. Richardt, and J. Tompkin. MatryODShka: Real-time 6DoF video view synthesis using multi-sphere images. In *European Conference on Computer Vision (ECCV)*, Aug. 2020.

[3] T. Bertel, N. D. F. Campbell, and C. Richardt. Megaparallax: Casual 360° panoramas with motion parallax. *IEEE Transactions on Visualization and Computer Graphics*, 25(5):1828–1835, 2019. doi: 10.1109/TVCG.2019.2898799

[4] T. Bertel, M. Yuan, R. Lindroos, and C. Richardt. OmniPhotos: Casual 360° VR photography. *ACM Transactions on Graphics*, 39(6):266:1–12, Dec. 2020. doi: 10.1145/3414685.3417770

[5] M. Broxton, J. Flynn, R. Overbeck, D. Erickson, P. Hedman, M. Duvall, J. Dourgarian, J. Busch, M. Whalen, and P. Debevec. Immersive light field video with a layered mesh representation. *ACM Trans. Graph.*, 39(4), July 2020. doi: 10.1145/3386569.3392485

[6] Z. Cao, A. Kar, C. Hane, and J. Malik. Learning independent object motion from unlabelled stereoscopic videos. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[7] A. Chang, A. Dai, T. Funkhouser, M. Halber, M. Niessner, M. Savva, S. Song, A. Zeng, and Y. Zhang. Matterport3d: Learning from RGB-D data in indoor environments. *International Conference on 3D Vision (3DV)*, 2017.

[8] G. Chaurasia, S. Duchene, O. Sorkine-Hornung, and G. Drettakis. Depth synthesis and local warps for plausible image-based navigation. *ACM Transactions on Graphics (TOG)*, 32(3):1–12, 2013.

[9] X. Chen, J. Song, and O. Hilliges. Monocular neural image based rendering with continuous view control. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2019.

[10] H. Cho, J. Kim, and W. Woo. Novel view synthesis with multiple 360 images for large-scale 6-dof virtual reality system. In *2019 IEEE Conference on Virtual Reality and 3D User Interfaces (VR)*, pp. 880–881, 2019. doi: 10.1109/VR.2019.8798142

[11] P. Debevec, G. Downing, M. Bolas, H.-Y. Peng, and J. Urbach. Spherical light field environment capture for virtual reality using a motorized pan/tilt

head and offset camera. In *ACM SIGGRAPH 2015 Posters*, SIGGRAPH '15. Association for Computing Machinery, New York, NY, USA, 2015. doi: 10.1145/2787626.2787648

[12] H. Dhamo, K. Tateno, I. Laina, N. Navab, and F. Tombari. Peeking behind objects: Layered depth prediction from a single image. *Pattern Recognition Letters*, 125:333 – 340, 2019. doi: 10.1016/j.patrec.2019.05.007

[13] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun. CARLA: An open urban driving simulator. In *Proceedings of the 1st Annual Conference on Robot Learning*, pp. 1–16, 2017.

[14] R. Du, D. Li, and A. Varshney. Geollery: A Mixed Reality Social Media Platform. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*, number 685. ACM, May 2019. doi: 10.1145/3290605.3300915

[15] R. Du, D. Li, and A. Varshney. Project Geollery.com: Reconstructing a Live Mirrored World With Geotagged Social Media. In *Proceedings of the 24th International Conference on Web3D Technology*, pp. 1–9. ACM, Jul. 2019. doi: 10.1145/3329714.3338126

[16] R. Du, E. Turner, M. Dzitsiuk, L. Prasso, I. Duarte, J. Dourgarian, J. Afonso, J. Pascoal, J. Gladstone, N. Cruces, S. Izadi, A. Kowdle, K. Tsotsos, and D. Kim. DepthLab: Real-Time 3D Interaction With Depth Maps for Mobile Augmented Reality. In *Proceedings of the 33rd Annual ACM Symposium on User Interface Software and Technology*, UIST, pp. 829–843. ACM, Oct. 2020. doi: 10.1145/3379337.3415881

[17] B. Y. Feng, W. Yao, Z. Liu, and A. Varshney. Deep depth estimation on 360° images with a double quaternion loss. In *2020 International Conference on 3D Vision (3DV)*, pp. 524–533, 2020. doi: 10.1109/3DV50981.2020.00062

[18] J. Flynn, M. Broxton, P. Debevec, M. DuVall, G. Fyffe, R. Overbeck, N. Snavely, and R. Tucker. DeepView: View synthesis with learned gradient descent. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[19] J. Flynn, I. Neulander, J. Philbin, and N. Snavely. DeepStereo: Learning to predict new views from the world's imagery. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 5515–5524, 2016.

[20] C. Fruh and A. Zakhor. Constructing 3D city models by merging aerial and ground views. *IEEE Computer Graphics and Applications*, 23(6):52–61, 2003.

[21] R. Hartley and A. Zisserman. *Multiple view geometry in computer vision*. Cambridge University Press, 2003.

[22] P. Hedman, S. Alsisan, R. Szeliski, and J. Kopf. Casual 3D Photography. *ACM Transactions on Graphics (Proc. SIGGRAPH Asia)*, 36(6):234:1–234:15, 2017.

[23] P. Hedman and J. Kopf. Instant 3D Photography. *ACM Transactions on Graphics (Proc. SIGGRAPH)*, 37(4):101:1–101:12, 2018.

[24] P. Hedman, J. Philip, T. Price, J.-M. Frahm, G. Drettakis, and G. Brostow. Deep blending for free-viewpoint image-based rendering. *ACM Transactions on Graphics (Proc. SIGGRAPH Asia)*, 37(6):257:1–257:15, 2018.

[25] N. Hobloss, A. Purica, A. Fiandrotti, M. Cagnazzo, R. Cozot, and W. Hamidouche. A hybrid approach to wide baseline view synthesis with convolutional neural networks. In *2019 International Conference on 3D Immersion (IC3D)*, pp. 1–7, 2019. doi: 10.1109/IC3D48390.2019.8976000

[26] J. Huang, Z. Chen, D. Ceylan, and H. Jin. 6-DOF VR videos with a single 360-camera. In *2017 IEEE Virtual Reality (VR)*, pp. 37–44, 2017. doi: 10.1109/VR.2017.7892229

[27] S. Im, H. Ha, F. Rameau, H.-G. Jeon, G. Choe, and I. S. Kweon. All-around depth from small motion with a spherical panoramic camera. In B. Leibe, J. Matas, N. Sebe, and M. Welling, eds., *Computer Vision – ECCV 2016*, pp. 156–172. Springer International Publishing, Cham, 2016.

[28] S. Khamis, S. Fanello, C. Rhemann, A. Kowdle, J. Valentin, and S. Izadi. Stereonet: Guided hierarchical refinement for real-time edge-aware depth prediction. In *Proceedings of the European Conference on Computer Vision (ECCV)*, September 2018.

[29] J. Kopf, K. Matzen, S. Alsisan, O. Quigley, F. Ge, Y. Chong, J. Patterson, J.-M. Frahm, S. Wu, M. Yu, et al. One shot 3D photography. *ACM Transactions on Graphics (TOG)*, 39(4):76–1, 2020.

[30] Z. Li, Z. Cui, M. R. Oswald, and M. Pollefeys. Street-view panoramic video synthesis from a single satellite image, 2020.

[31] K.-E. Lin, Z. Xu, B. Mildenhall, P. P. Srinivasan, Y. Hold-Geoffroy, S. DiVerdi, Q. Sun, K. Sunkavalli, and R. Ramamoorthi. Deep multi depth panoramas for view synthesis, 2020.

[32] R. Liu, J. Lehman, P. Molino, F. Petroski Such, E. Frank, A. Sergeev, and J. Yosinski. An intriguing failing of convolutional neural networks and the coordconv solution. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, eds., *Advances in Neural Information Processing Systems 31*, pp. 9605–9616. Curran Associates, Inc., 2018.

[33] X. Lu, Z. Li, Z. Cui, M. R. Oswald, M. Pollefeys, and R. Qin. Geometry-aware satellite-to-ground image synthesis for urban areas. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[34] B. Luo, F. Xu, C. Richardt, and J. Yong. Parallax360: Stereoscopic 360° scene representation for head-motion parallax. *IEEE Transactions on Visualization and Computer Graphics*, 24(4):1545–1553, 2018. doi: 10.1109/TVCG.2018.2794071

[35] R. Martin-Brualla, N. Radwan, M. S. Sajjadi, J. T. Barron, A. Dosovitskiy, and D. Duckworth. Nerf in the wild: Neural radiance fields for unconstrained photo collections. *arXiv preprint arXiv:2008.02268*, 2020.

[36] B. Mildenhall, P. P. Srinivasan, M. Tancik, J. T. Barron, R. Ramamoorthi, and R. Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. In *ECCV*, 2020.

[37] K. Mueller, A. Smolic, K. Dix, P. Merkle, P. Kauff, and T. Wiegand. View synthesis for advanced 3D video systems. *EURASIP Journal on Image and Video Processing*, 2008:1–11, 2009.

[38] E. Penner and L. Zhang. Soft 3D reconstruction for view synthesis. *ACM Trans. Graph.*, 36(6), Nov. 2017. doi: 10.1145/3130800.3130855

[39] N. Ravi, J. Reizenstein, D. Novotny, T. Gordon, W.-Y. Lo, J. Johnson, and G. Gkioxari. Accelerating 3D deep learning with pytorch3d. *arXiv:2007.08501*, 2020.

[40] O. Ronneberger, P. Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pp. 234–241. Springer, 2015.

[41] M. Savva, A. Kadian, O. Maksymets, Y. Zhao, E. Wijmans, B. Jain, J. Straub, J. Liu, V. Koltun, J. Malik, D. Parikh, and D. Batra. Habitat: A Platform for Embodied AI Research. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019.

[42] J. L. Schönberger and J.-M. Frahm. Structure-from-motion revisited. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[43] J. L. Schönberger, E. Zheng, M. Pollefeys, and J.-M. Frahm. Pixelwise view selection for unstructured multi-view stereo. In *European Conference on Computer Vision (ECCV)*, 2016.

[44] S. Schubert, P. Neubert, J. Pöschmann, and P. Protzel. Circular convolutional neural networks for panoramic images and laser data. In *2019 IEEE Intelligent Vehicles Symposium (IV)*, pp. 653–660, 2019. doi: 10.1109/IVS.2019.8813862

[45] A. Serrano, I. Kim, Z. Chen, S. DiVerdi, D. Gutierrez, A. Hertzmann, and B. Masia. Motion parallax for 360° RGBD video. *IEEE Transactions on Visualization and Computer Graphics*, 25(5):1817–1827, 2019. doi: 10.1109/TVCG.2019.2898757

[46] Y. Shi, D. Campbell, X. Yu, and H. Li. Geometry-guided street-view panorama synthesis from satellite imagery, 2021.

[47] M.-L. Shih, S.-Y. Su, J. Kopf, and J.-B. Huang. 3D photography using context-aware layered depth inpainting. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[48] P. P. Srinivasan, R. Tucker, J. T. Barron, R. Ramamoorthi, R. Ng, and N. Snavely. Pushing the boundaries of view extrapolation with multiplane images. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[49] Y.-C. Su, D. Jayaraman, and K. Grauman. Pano2vid: Automatic cinematography for watching 360 videos. In *Asian Conference on Computer Vision*, pp. 154–171. Springer, 2016.

[50] Y. Sun, A. Lu, and L. Yu. Weighted-to-spherically-uniform quality evaluation for omnidirectional video. *IEEE Signal Processing Letters*, 24(9):1408–1412, 2017.

[51] T. Teo, L. Lawrence, G. A. Lee, M. Billinghurst, and M. Adcock. Mixed reality remote collaboration combining 360 video and 3D reconstruction. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*, pp. 1–14, 2019.

[52] R. Tucker and N. Snavely. Single-view view synthesis with multiplane images. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[53] S. Tulsiani, R. Tucker, and N. Snavely. Layer-structured 3D scene inference via view synthesis. In *Proceedings of the European Conference on*

Computer Vision (ECCV), September 2018.

[54] S. Ullman. The interpretation of structure from motion. *Proceedings of the Royal Society of London. Series B. Biological Sciences*, 203(1153):405–426, 1979.

[55] N. H. Wang, B. Solarte, Y. H. Tsai, W. C. Chiu, and M. Sun. 360sd-net: 360 stereo depth estimation with learnable cost volume. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 582–588, 2020.

[56] O. Wiles, G. Gkioxari, R. Szeliski, and J. Johnson. SynSin: End-to-end view synthesis from a single image. In *CVPR*, 2020.

[57] C. Won, J. Ryu, and J. Lim. Sweepnet: Wide-baseline omnidirectional depth estimation. In *2019 International Conference on Robotics and Automation (ICRA)*, pp. 6073–6079, 2019.

[58] J. Xiao, T. Fang, P. Zhao, M. Lhuillier, and L. Quan. Image-based street-side city modeling. *ACM Transactions on Graphics (TOG)*, 28(5):114, 2009.

[59] J. Xiao and Y. Furukawa. Reconstructing the world's museums. *International journal of computer vision*, 110(3):243–258, 2014.

[60] K. Zhang, G. Riegler, N. Snavely, and V. Koltun. Nerf++: Analyzing and improving neural radiance fields, 2020.

[61] Y. Zhang and T. Funkhouser. Deep depth completion of a single rgb-d image. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[62] H. Zhao, O. Gallo, I. Frosio, and J. Kautz. Loss functions for image restoration with neural networks. *IEEE Transactions on Computational Imaging*, 3(1):47–57, 2017. doi: 10.1109/TCI.2016.2644865

[63] Q. Zhao, L. Wan, W. Feng, J. Zhang, and T.-T. Wong. Cube2video: Navigate between cubic panoramas in real-time. *IEEE Transactions on Multimedia*, 15(8):1745–1754, 2013. doi: 10.1109/TMM.2013.2280249

[64] T. Zhou, R. Tucker, J. Flynn, G. Fyffe, and N. Snavely. Stereo magnification: Learning view synthesis using multiplane images. In *SIGGRAPH*, 2018.

[65] N. Zioulis, A. Karakottas, D. Zarpalas, F. Alvarez, and P. Daras. Spherical view synthesis for self-supervised 360^o depth estimation. In *International Conference on 3D Vision (3DV)*, September 2019.

[66] N. Zioulis, A. Karakottas, D. Zarpalas, and P. Daras. Omnidepth: Dense depth estimation for indoors spherical panoramas. In *Proceedings of the European Conference on Computer Vision (ECCV)*, September 2018.