3D PANO INPAINTING: SCENE CONSTRUCTION USING A SINGLE INPUT PANORAMA

A Thesis

presented to

the Faculty of California Polytechnic State University,

San Luis Obispo

In Partial Fulfillment

of the Requirements for the Degree

Master of Science in Computer Science

by

Shivam Asija

March 2024
COMMITTEE MEMBERSHIP

TITLE: 3D Pano Inpainting: Scene Construction Using a Single Input Panorama

AUTHOR: Shivam Asija

DATE SUBMITTED: March 2024

COMMITTEE CHAIR: Jonathan Ventura, Ph.D.
Professor of Computer Science

COMMITTEE MEMBER: Sumona Mukhopadhyay, Ph.D.
Professor of Computer Science

COMMITTEE MEMBER: John Seng, Ph.D.
Professor of Computer Science
ABSTRACT

3D Pano Inpainting: Scene Construction Using a Single Input Panorama

Shivam Asija

Creating 360-degree 3D content has gained traction in the past few years, being used for Virtual Reality environments. However, creating such content is challenging because it requires a multi-camera setup or a collection of images from different perspectives. This paper proposes 3D Pano Inpainting, a pipeline capable of transforming a single equirectangular panoramic RGBD image into a complete 360° 3D virtual reality scene represented as a textured mesh. Our methodology is as follows: we estimate a consistent depth map for the input panorama; we use a pre-built framework to convert the image and its depth map into a textured mesh with inpainted background edges; we account for wrapping the resulting mesh around the viewer’s perspective for better immersion in VR headsets. Additionally, we evaluate our method’s effectiveness in producing consistent novel views through the peak signal-to-noise ratio (PSNR), structural similarity index (SSIM), and learned perceptual image patch similarity (LPIPS) between a rendering produced from the ground truth image and depth map to that produced from our model. Furthermore, we compare our model’s scores with those of a non-inpainted textured mesh.
ACKNOWLEDGMENTS

Thanks to:

- My parents, for supporting me throughout my academic career. I wouldn’t be here without you.
- Dr. Ventura, for allowing me to work on his VR project and for teaching me how to perform research
- My professors, for the invaluable knowledge throughout my academics
- My friends. Especially Darrick Oliver, for letting me convince him to pursue higher education so I can have a buddy in my classes, and Nilay Patel, for convincing me to pursue a Master’s degree at Cal Poly
- Edward Du, for helping me with the evaluation section of this paper
- Nam Nguyen, for helping produce the figures in this paper
# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>LIST OF TABLES</th>
<th>ix</th>
</tr>
</thead>
<tbody>
<tr>
<td>LIST OF FIGURES</td>
<td>x</td>
</tr>
</tbody>
</table>

## CHAPTER

1. Introduction .................................................. 1

2. Background .................................................... 4
   - 2.1 Depth Estimation .................................... 4
     - 2.1.1 360 Monodepth ..................................... 4
       - 2.1.1.1 MiDaS ........................................... 6
       - 2.1.1.2 Global disparity map alignment ............... 7
       - 2.1.1.3 Disparity map blending ....................... 9
     - 2.2 View synthesis ..................................... 10
       - 2.2.1 3d-photo inpainting ............................ 10
         - 2.2.1.1 Preprocessing ................................ 11
         - 2.2.1.2 Identifying context and synthesis regions ... 12
         - 2.2.1.3 Context-aware color and depth inpainting ... 13
         - 2.2.1.4 Adversarial Networks ....................... 14
         - 2.2.1.5 Converting to 3D textured mesh ............. 15

3. Related Works .................................................. 16
   - 3.1 Perspective view synthesis ........................... 16
     - 3.1.1 Encoder-decoder .................................. 16
     - 3.1.2 Differentiable point cloud renderer ............ 17
     - 3.1.3 Neural radiance fields ........................... 18
3.1.4 CNNs for Multi-plane images ........................................ 19

3.2 Panoramic View Synthesis ............................................. 20
   3.2.1 Virtual Environment from Panoramic View .................... 20
   3.2.2 Multi-depth panoramas ......................................... 21
   3.2.3 CNNs .......................................................... 22
   3.2.4 Multi-cylindrical images ...................................... 23

4 Implementation ............................................................ 24
   4.1 Input ............................................................. 24
   4.2 Depth Estimation .................................................. 25
      4.2.1 Boost-monodepth + wrap padding .......................... 25
      4.2.2 360Monodepth ............................................. 26
   4.3 Building Meshes ................................................... 27
   4.4 Coordinate Projection ............................................. 28
   4.5 View Synthesis ................................................... 29

5 Experimental Setup ..................................................... 30
   5.1 Replica Dataset .................................................. 31
      5.1.1 Generate Viewpoints ...................................... 31
      5.1.2 Camera poses ............................................ 32
      5.1.3 Ground Truth Depth ....................................... 32
   5.2 Image Metrics .................................................... 33

6 Results ................................................................. 34
   6.1 Quantitative Evaluation .......................................... 34
   6.2 Qualitative Evaluation .......................................... 37
   6.3 Discussion ........................................................ 38

7 Future Works ............................................................ 41
## LIST OF TABLES

<table>
<thead>
<tr>
<th>Table</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>6.1</td>
<td>Results of quantitative evaluation on the synthetic scene &quot;hotel 0&quot;. Values are the averages of the three renderings per viewpoint. The best value of each metric is shown in bold.</td>
<td>34</td>
</tr>
<tr>
<td>6.2</td>
<td>Results of quantitative evaluation on the synthetic scene &quot;office 0&quot;. Values are the averages of the three renderings per viewpoint. The best value of each metric is shown in bold.</td>
<td>35</td>
</tr>
<tr>
<td>6.3</td>
<td>Results of quantitative evaluation on the synthetic scene &quot;room 0&quot;. Values are the averages of the three renderings per viewpoint. The best value of each metric is shown in bold.</td>
<td>36</td>
</tr>
<tr>
<td>Figure</td>
<td>Page</td>
<td></td>
</tr>
<tr>
<td>--------</td>
<td>------</td>
<td></td>
</tr>
<tr>
<td>1.1</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>Illustration of the Six Degrees of Freedom[1]. This diagram represents the three-dimensional space motion possibilities: translational (up/down, left/right, and forward/backward) and rotational motion (roll, pitch, and yaw).</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2.1</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>Each icosahedron face (blue triangle) is fit inside a rectangular cropping of the perspective projection of the equirectangular panorama (blue rectangle). [19] The padding are as follows: p=0 (blue), p=0.1 (green), p=0.2 (red)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2.2</td>
<td>9</td>
<td></td>
</tr>
<tr>
<td>Comparison of different blending methods in Rey-Area et al.’s paper for tangent images[19]. Nearest neighbor ('NN') weights select estimates only from the nearest tangent image. Mean weights average out the overlapping tangent images per pixel. Radial weights decay at 15° from the center of projection. Frustum weights decay 30% diagonally from each corner. The arrows show areas where visible seams in the depth map are still present. 'Frustum' has the best results here.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2.3</td>
<td>11</td>
<td></td>
</tr>
<tr>
<td>Shih et al. preprocess their images by doing the following: (a-d) use a bi-lateral filter across the depth map using a $7 \times 7$ window size to sharpen blurred pixels and precisely localize depth discontinuities; (e) detect discontinuities by thresholding disparity differences between pixels; (f) clean up and link discontinuities into connected depth edges[22].</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2.4</td>
<td>13</td>
<td></td>
</tr>
<tr>
<td>This shows the process of synthesizing pixels using inpainting. (a) the mesh is fully connected. (b) mesh is disconnected across discontinuities, resulting in the foreground edge (green) and the background edge (red). (c) For the background layer, a context (blue) and a synthesis region (red) need to be produced. (d) Finally, the synthesis pixels are merged into the LDI. [22]</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
2.5 Shih et al. take the color, depth, and linked depth edges as input and randomly pick an edge as a subproblem. They use an edge inpainting network to fill in the depth edge within the synthesis region. Next, they use a color inpainting network on the new inpainted depth edges with the contextual color to generate the inpainted color. Similarly, they use a depth inpainting network on the same depth edges with the contextual depth to synthesize the inpainted depth. [22]

4.1 With a single input panorama and a depth map generated by 360MonoDepth[19], our system uses a combination of inpainting and spherical projection to generate (1) a 3D mesh that is capable of VR rendering, and (2) novel panorama views [22].

4.2 Example inpainting result from our system. (a) The black regions are gaps in the mesh caused by depth discontinuities. (b) The gaps have been filled with mesh geometry and inpainted.

6.1 Qualitative comparisons for room 0

6.2 Comparison of a textured mesh model (inset bottom) and our Panopainting with 360MonoDepth (inset top).

6.3 Illustration of the conversion between a cubemap and an equirectangular projection. [24]
Chapter 1

INTRODUCTION

The focus on virtual reality (VR) intensified in recent years because of the pandemic, as VR can be seen as a viable solution to the challenges faced with online education. Consequently, large tech companies like Meta, and now Apple, have invested millions into producing consumer-grade VR headsets with a plethora of applications. Such availability has led to an increased exploration of immersive 360° images.

360° cameras like the Insta One RS or Theta X provide the capability for conveniently capturing 360 images and videos by casual as well as professional content creators. These images can be experienced in a 360° format on VR headset devices. While the 360° format allows users to explore the photograph in an immersive way by filling the users’ field of view, they lack support for motion parallax during translational head movements, resulting in an unnatural experience that may disrupt immersion and induce discomfort or even nausea in some users. Motion parallax is an occurrence where objects closer to a camera move faster than those that are far away and it can give the viewer a sense of depth perception. In the context of 360° images, under a head movement, objects closer to the viewer undergo a larger displacement as compared to objects far away [16]. The success of an immersive environment involves supporting a full 6 Degree-of-Freedom (6DoF) experience which allows users to have the ability to move freely in six directions: forward/backward, up/down, left/right, as well as yaw, pitch, and roll. In order to support 6DoF, earlier works involved using depth estimation to create 360° RGBD images. Although this did give them a sense of depth, user movements either exposed gaps or a stretching effect in areas surrounding foreground objects, negatively affecting the realistic depiction of these
Research in view synthesis helped address these areas, ultimately improving the VR experience.

View synthesis refers to the process of generating an image or video that depicts a scene from a specific viewpoint. The significance of view synthesis lies in its ability to better understand the scene’s geometry, enhance its visual quality, and support the creation of new views from existing data. Novel view synthesis is an extension of that, where the generated viewpoint differs from the original input. In most cases of novel view synthesis, the input provides context information about the scene with which an accurate prediction of a different viewpoint can be made. The goal is to create new views that are coherent and consistent with the original data. Novel view synthesis helps tackle the inconsistencies with the aforementioned RGBD images because we can use the information from our initial point-of-view image to fill the gaps in areas shown when the viewer’s perspective changes.
In this thesis, we look to perform novel view synthesis on 360° images and build immersive meshes that can be viewed in a virtual reality headset. Our method aims for day-to-day users, so we focus on using a single equirectangular panoramic image capable of being produced from a standard handheld device. An equirectangular panorama is useful as it encompasses a complete 360° field of view, both horizontally and vertically, allowing for a fully spherical representation of the environment as opposed to a cylindrical panorama, which only supports 360° in horizontal movement. Utilizing a single panorama is beneficial because of its ease of use computationally. Conversely, videos have too many frames to process, whereas, a single perspective image has too little information for VR-applicable view synthesis.

3D pano inpainting addresses the limited immersive experience of 360° images and 360° RGB-D images within a VR headset. We aim to use inpainting on single equirectangular panoramic images to create natural VR environments that support full 6DoF experience and solve issues related to gaps and stretching effects seen with RGB-D images. Our contributions include: (1) producing a mesh supporting 360° view synthesis with motion parallax, all from a single equirectangular panorama; and (2) assessing the quality of images rendered by our method using a synthetic dataset, employing PSNR, SSIM, and LPIPS image quality metrics. Although visually our method excelled in creating novel views with little to no stretching or gaps, quantitatively, our results fell short of expectations. This was likely due to misalignment between rendered views, which negatively impacted the scores.

One thing to note: throughout this paper, we use the terms '360° images' and 'panoramas' interchangeably, as well as 'equirectangular' and 'spherical.' Equirectangular projections are a type of spherical projection, and 360° images are a form of panoramas.
Our 3D Pano Inpainting pipeline is built on 2 prior research papers. Specifically, we use 360Monodepth’s high-resolution depth estimator[19] and 3D photo inpainting’s edge detection and inpainting methodologies [22]. The following sections give an in-depth explanation of their workings and thus are necessary to fully understand this thesis.

2.1 Depth Estimation

Depth estimation is the most important component in creating an immersive photograph because it extends an image from 2D to 3D space. Monocular depth estimation is a subtopic of depth estimation that requires a single view input image.

2.1.1 360 Monodepth

Earlier works of monocular depth estimation are limited to perspective and low-resolution images whereas Rey-Area et. al’s approach is made for equirectangular and high-resolution images[19]. In addition to this, their method is flexible in that the depth estimation algorithm can be swapped out to the current state-of-the-art for perspective images.

The first part of the 360Monodepth framework relies on breaking the 360° equirectangular panorama into a set of independent perspective tangent images. The reason for this is to deal with any distortions in equirectangular projections. Simply applying a
Each icosahedron face (blue triangle) is fit inside a rectangular cropping of the perspective projection of the equirectangular panorama (blue rectangle). [19] The padding are as follows: $p=0$ (blue), $p=0.1$ (green), $p=0.2$ (red)

Perspective depth estimator to an equirectangular image will ultimately give incorrect depth because equirectangular projections distort the spatial relationships and scales of objects, especially towards the poles of the image where the stretching becomes more obvious. This distortion can significantly skew depth perception, leading to inaccurate depth estimations. The standard for the 360Monodepth framework involves extracting 20 perspective rectangular images from the panoramic view, each corresponding to the face of an icosahedron that encircles a sphere. The sphere represents the ideal depth map for spherical panoramas. The triangular faces are tangent to the sphere at its center and they can now be independently used by any perspective depth estimating algorithm to produce a collection of 20 depth maps. A thing to point out is that there will be some minor overlap between connecting icosahedron faces since they are cropping out a rectangular image for a triangular face. Doing such allows for smoother continuity and a consistent alignment of the depths after the overlapping sections are blended. In addition to an overlap, Rey-Area et al. also use a padding factor on the rectangular cropping for smoother alignment. For example, a $p=0.3$ is a 30% padding on each side of the rectangular cropping.
To perform monocular depth estimation on each of the perspective images, Rey-Area et al. use MiDaS v2 [18] and v3 [17] because of their performance in both indoor and outdoor scenes. MiDaS v2 is trained on a variety of different datasets with diverse depth sensing approaches (stereo cameras, laser scanners, and light sensors). V2’s novelty lies in its loss functions that are immune to incompatibilities between datasets. MiDaS v3 introduces an entirely new architecture in place of convolutional neural networks (CNNs): the vision transformer[5]. CNN-based predictors’ architecture consists of an encoder and decoder. The encoder extracts necessary features similar to an image classification network, whereas the decoder predicts the depth based on the features collected from the encoder. Convolution encoders downsample the input image for feature extraction, which means that there will be some loss of information: feature resolution and granularity in this case. Using vision transformer encoders, on the other hand, forgoes excessive downsampling and maintains a constant dimensionality throughout all processing stages. This is advantageous for dense prediction as granular details are preserved.

MiDaS produces disparity maps that represent the inverse of depth, but due to its scale and shift invariant training approach, the specific scale factor and shift offset remain unknown. Perspective disparity maps by MiDaS are orthogonal to the camera’s viewing direction, but each image needs to be at a different viewing direction to conform to the equirectangular projection. Additionally, spherical disparity is defined as the reciprocal of the Euclidean distance from the camera’s center to a 3D point.

\[
\text{spherical disparity} = \frac{1}{\sqrt{\sum_{i=1}^{n}(x_i - y_i)^2}}
\]  

(2.1)
After generating the disparity maps from MiDaS, the next steps involve converting them from the perspective to spherical disparity and from the tangent image space to the equirectangular projection space.

2.1.1.2 Global disparity map alignment

As previously noted, the MiDaS estimator yields disparity maps with inconsistent scales and offsets, necessitating global alignment. Rey-Area et al.’s approach to disparity alignment employs spatially diverse affine adjustment fields [19] instead of a constant scale and offset per disparity map. These fields are represented as 2D grids in tangent image space with dimensions $m \times n$, where each grid element $i$ holds a pair of scale and offset parameters $(s_i, o_i)$ that are interpolated bilinearly (extrapolating the intermediate value based on surrounding pixels) across the tangent image. The bilinear interpolation is defined by the following:

$$\tilde{D}(x) = s(x)D(x) + o(x),$$

(2.2)

where $o(x) = \sum_i w_i(x) o_i$ and $s(x) = \sum_i w_i(x) s_i$ are the intermediate offset and scale values, and $w_i(x)$ are the bilinear interpolation weights at pixel location $x$.

The global alignment follows an optimization that targets the affine adjustment fields which minimizes the combined energy function:

$$\arg\min_{\{s_a, o_a\}} E_{\text{alignment}} + \lambda_{\text{smoothness}} E_{\text{smoothness}} + \lambda_{\text{scale}} E_{\text{scale}}$$

(2.3)

Here the tradeoff is between the alignment and the spatial smoothness of adjustment fields in conjunction with a scale regularisation term. They use $\lambda_{\text{smoothness}} = 40$ and
\[ \lambda_{\text{scale}} = 0.007 \] for all instances of the optimization. The terms in the optimization function are described as follows:

- **Disparity alignment term.** Given the set \( Z = \{(a, b) | a, b \in \mathcal{T}, a < b\} \) of ordered pairs of tangent images, where \( \mathcal{T} \) is a set of tangent image indices, and the set of overlapping pixels in images \( a \) and \( b \) is denoted by \( \Omega(a, b) \), the alignment between rescaled disparity maps \( \tilde{D}_a \) and \( \tilde{D}_b \) is:

\[
E_{\text{alignment}} = \frac{1}{z_a} \sum_{(a, b) \in Z} \sum_{x \in \Omega(a, b)} \left( \tilde{D}_a(x) - \tilde{D}_b(x) \right)^2 \tag{2.4}
\]

where \( z_a = \sum_{(a, b) \in Z} |\Omega(a, b)| \) is a normalizing term dependent on the number of overlapping pixel pairs. The disparity alignment term ensures an agreement in overlapping sections of the disparity maps. The higher the alignment term, the less of an overlap between disparities.

- **Smoothness term.** The smoothness term allows neighboring grid points \( i \) and \( j \) within the adjustment fields to be spatially smooth as denoted by:

\[
E_{\text{smoothness}} = \frac{1}{z_s} \sum_{a \in \mathcal{T}} \sum_{(i, j)} \left( ||s^i_a - s^j_a||^2 + ||o^i_a - o^j_a||^2 \right) \tag{2.5}
\]

where \( z_s = |\mathcal{T}| / m \ast n \) normalizes the term by the number of grid-points in all tangent images. A high smoothness term means that the scale and offset values within the grid are too high, resulting in highly varying scale and offset disparities.

- **Scale term.** The scale term regularises the scale to avoid a collapse of values to \( s = 0 \):

\[
E_{\text{scale}} = \sum_{a \in \mathcal{T}} \sum_i (s^i_a)^{-1} \tag{2.6}
\]
Figure 2.2: Comparison of different blending methods in Rey-Area et al.’s paper for tangent images[19]. Nearest neighbor (‘NN’) weights select estimates only from the nearest tangent image. Mean weights average out the overlapping tangent images per pixel. Radial weights decay at 15° from the center of projection. Frustum weights decay 30% diagonally from each corner. The arrows show areas where visible seams in the depth map are still present. ‘Frustum’ has the best results here.

To enhance the refinement process for global alignment, they implement optimization across various scales. Initially, they perform optimization for a grid consisting of 4 × 3 points per tangent disparity map, integrating adjustment fields into the disparity maps. This procedure is then repeated for grid sizes of 8 × 7 and 16 × 14, with adjustments applied to the maps after each optimization.

2.1.1.3 Disparity map blending

Once the alignment is done, the next step is to blend the disparity maps into a single spherical map. Figure 2.2 shows different blending methods, here are a few key points to make. Nearest neighbor and mean blending methods have very visible seams as shown by the arrows. Blending weights in the shape of a frustum leads to the least amount of seams but results can get blurry. For the highest quality blending, Rey-Area et al. blended the maps with Poisson blending which minimizes the following
optimization function:

\[
\arg\min_B \sum_{a \in \mathcal{T}} \sum_x \omega_a(x) \left\| \nabla B(x) - \nabla \tilde{D}_a(x) \right\|^2_2 + \lambda_{\text{fidelity}} \sum_x (B(x) - D_{\text{NN}}(x))^2
\]  (2.7)

This optimization produces a blended map \( B(\cdot) \). Here \( \omega_a(x) \) are the 'frustum' blending weights which adjust the effect of pixels and \( \lambda_{\text{fidelity}} = 0.1 \) allows the \( D_{\text{NN}} \) stitch to have some influence on the resulting blended disparity map.

### 2.2 View synthesis

Typically, noticeable gaps and distortions emerge in 3D photographs with naïve depth-based warping techniques, such as holes or stretching. Novel view synthesis addresses the challenge of such disocclusions arising from parallax effects.

#### 2.2.1 3d-photo inpainting

Shih et al. present a paper employing inpainting methodologies to mitigate these issues [22]. Their approach utilizes an RGB-D input, images with an aligned color and depth pair, to generate a Layered Depth Image (LDI) with inpainted color and depth in occluded regions. LDIs are images where there can be any number of pixels at a single position within their pixel lattice, differentiated only by depth differences. As a result, if an image has multiple pixels at different depths in the same lattice location, a parallax movement of the viewpoint will reveal the background pixels at that same location [21]. This allows for more memory-efficient handling of multiple layers since independent pixels at different layers can be stored within a single file. LDI pixels are connected across the 4 cardinal directions (left, right, top, bottom) but they are not connected across depth discontinuities.
Figure 2.3: Shih et al. preprocess their images by doing the following: (a-d) use a bi-lateral filter across the depth map using a $7 \times 7$ window size to sharpen blurred pixels and precisely localize depth discontinuities; (e) detect discontinuities by thresholding disparity differences between pixels; (f) clean up and link discontinuities into connected depth edges\cite{22}.

2.2.1.1 Preprocessing

The 3D inpainting method begins with the preprocessing stage, where they convert an RGB-D image into an LDI by connecting all pixels of the image with its neighbors in the four cardinal directions. Once that is done, they then find the depth discontinuities in the image, these are areas where the difference in depth between neighboring pixels exceeds a certain threshold. In most depth estimation methods, depth discontinuities are blurred across multiple pixels which makes them harder to identify. To account for any blurriness across pixels in discontinuity regions of the image, they apply an image filter using a $7 \times 7$ window size (Figure 2.3c-d). Once sharpened, discontinuity regions within the images can be found by thresholding the difference in disparities between neighboring pixels. In certain cases, there can be a small number of isolated splotches of pixels where there are depth discontinuities but there isn’t a connected edge, in which case some cleaning is needed (Figure 2.3e).

To clean the depth edges, Shih et al. employ the following: (1) they binary label discontinuities; (2) they use connected component analysis to merge adjacent discontinuities into a collection of linked edges; and (3) they remove any short segments (i
10 pixels). Because of their effectiveness in identifying connected discontinuity edges, we also employ these techniques into our method. Finally, the inpainting procedure can be applied to the remaining edge segments.

2.2.1.2 Identifying context and synthesis regions

Given one of the depth edge segments, the goal is to generate new pixels with color and depth next to the occluded region. This is done by first disconnecting the LDI pixels across the edges where a discontinuity was identified, resulting in a foreground and background edge as shown in figure 2.4b. Only the background edge is of interest for inpainting since the existing content surrounding the background silhouette is what is being extended.

Before starting the learning-based inpainting process, there needs to be a generation of the synthesis region and the identification of a context region. The synthesis region constitutes the newly created sections and it starts with just a set of 2D pixel coordinates that undergoes an iterative expansion process to initialize its color and depth values. This is accomplished by using a flood-fill algorithm from the background silhouette described as follows: firstly, it takes a single pixel step outward from the disconnected edge, establishing that as the initial synthesis region; then through a series of 40 iterations, pixels within the region are methodically expanded in all four cardinal directions (left, right, up, down) ensuring the inclusion of unvisited pixels. The expansion is between both the synthesis and context regions, where each pixel added belongs exclusively to one of the two regions. The context region provides the learning-based inpainting model with necessary information about existing pixels to generate colors and depth into the synthesis region. Identifying the context region also follows the same flood-fill algorithm, the only difference is that the iteration is
Figure 2.4: This shows the process of synthesizing pixels using inpainting. (a) the mesh is fully connected. (b) mesh is disconnected across discontinuities, resulting in the foreground edge (green) and the background edge (red). (c) For the background layer, a context (blue) and a synthesis region (red) need to be produced. (d) Finally, the synthesis pixels are merged into the LDI. [22]

across LDI pixels instead of the synthesis region. The context region expands 100 iterations beyond the disconnect edge for best inpainting performance.

Since depth values are blurred in areas of discontinuities, the synthesis region may not align well with the actual occluding boundaries. To mitigate this error, the synthesis region is dilated across the depth edge in the context region by 5 pixels. This dilation ensures a smooth transition of the synthesized pixels into the LDI.

2.2.1.3 Context-aware color and depth inpainting

Since the context and synthesis regions within the LDI are similar to images, image-based inpainting network architectures can still be used. The color and depth inpainting architecture is designed as follows: (1) edge inpainting network, (2) color inpainting network, and (3) depth inpainting network as shown in figure 2.5. The edge inpainting network takes the context edges as an input and predicts the depth edges in the synthesis region, producing the inpainted edges and concatenating them to the LDI. This allows the architecture to first infer the structure of the area where content prediction (color and depth extrapolation) will occur. With the inferred edge and the context region’s color as inputs, the color inpainting network produces the
Figure 2.5: Shih et al. take the color, depth, and linked depth edges as input and randomly pick an edge as a subproblem. They use an edge inpainting network to fill in the depth edge within the synthesis region. Next, they use a color inpainting network on the new inpainted depth edges with the contextual color to generate the inpainted color. Similarly, they use a depth inpainting network on the same depth edges with the contextual depth to synthesize the inpainted depth. [22]

Inpainted colors for the pixels. Similarly, the depth inpainting network uses the concatenated edge and context depth to produce the inpainted depth values. In scenarios where the depth is complex, simply applying the inpainting a single time is insufficient as the edge model creates discontinuities that result in holes. Thus the model is repeated until no further depth edges can be generated.

2.2.1.4 Adversarial Networks

Inpainting networks follow an adversarial model which consists of a discriminator (D) and a generator (G)[14]. Discriminative models map a high-dimensional, complex input such as an image to a class label. Generative models create data that resembles the training data. They play a minimax game where G generates data and D probabilistically predicts whether the data was given from the training set or G. G’s goal is to maximize the probability of fooling D. Since both G and D are neural
networks, they can both be trained through backpropagation[8]. In the case of inpainting, the discriminators consist of encoders that downsample the images, whereas the generators consist of decoders that upsample images back to normal size.

The inpainting architecture is trained from a synthetic dataset. The process to generate the training data is as follows: first, obtain pseudo ground truth depth maps on the MSCOCO dataset [11] using a pre-trained MegaDepth [9] model; then, extract context and synthesis regions using the method described earlier; finally, randomly sample and place these context-synthesis regions on different images in the dataset. The ground truth content (RGB-D) is already obtained for these regions from the input image and depth pair. All three models within the architecture are trained through the minimax game mentioned earlier: each model has a discriminator/generator pair. All three generators within the models are trained on the context-synthesis region dataset.

2.2.1.5 Converting to 3D textured mesh

The final 3D triangular mesh is created when all of the inpainted depth and color values are appended onto the original LDI. This happens through an iterative process where the color data within the LDI is appended to 3-dimensional mesh vertices, and those vertices are connected with their neighbors in a triangular shape. The vertices’ position (X, Y, Z) values are determined by coordinate projection. A triangle face’s color within the mesh is a blend of the colors of its containing vertices. The resulting 3D representation can be easily rendered using basic graphic engines on edge devices.
View synthesis, the process of generating a new image or video that depicts a scene from a different viewpoint, has been a recurring problem that Computer Vision researchers have aimed to solve. As a result, there have been different successful avenues to achieve view synthesis. In this section, we examine the diverse techniques for achieving view synthesis. Following that, we refine our focus to explore view synthesis specifically within the domain of 360° images.

3.1 Perspective view synthesis

This section outlines some state-of-the-art research in the field of view synthesis. Although these methods work exceptionally well for perspective images, there has not been an extension to them to accommodate 360° views.

3.1.1 Encoder-decoder

One such method involves using an encoder and a decoder. This method by Zhou et al. is based on the idea that the appearance (texture, shape, color, etc.) of novel views are correlated and input images have enough information to infer views without the need for additional appearance synthesis[30]. The encoder-decoder method refers to a deep generative convolutional model which produces appearance flow vectors that explicitly deduce the appearance correlation between different views of the subject. Consequently, this learned-based approach implicitly depicts the structure
of the scene. For each output pixel, appearance flow vectors indicate the corresponding pixel in the input view to copy from. More specifically, the model takes (1) an input view and (2) a view transformation and encodes them via several convolutions and fully connected layers to extract relevant features of the environment. The decoder then assembles the features and outputs an appearance flow field (AFF) using two fully connected layers and six up-sampling convolution layers. The AFF, along with the input view, yields the target view via bilinear sampling. Like all network architectures, the layer weights are learned through back-propagation.

Although this approach demonstrated impressive results in object view synthesis, its effectiveness was limited when dealing with complex scenes due to the model’s inability to extrapolate pixel values that were absent in the input image. 3D Pano Inpainting works well for complex scenes as we are (1) able to extract very high-resolution depth maps and (2) our learned-based inpainting can generate new pixels when needed.

3.1.2 Differentiable point cloud renderer

SynSin is a model for view synthesis from a single image in complex scenes[27]. It produces a 3D scene using a high-resolution point cloud of learned features from the input image \(I\) and a new pose \(T\). The architecture consists of a spatial feature predictor which learns a set of features for higher level representation of the scene and a depth regressor which learns a depth map for the 3D structure of the input image. The spatial feature predictor is based on adversarial network architecture, whereas the depth regressor is based on an encoder-decoder architecture. They combined the spatial features and predicted depths to produce a 3D point cloud of features. Next, the point cloud is transformed according to the target viewpoint \(T\) and rendered. The rendered features are passed through a refinement network to produce the final
image. The refinement network’s role is to inpaint any missing regions in the rendering that arise from the transformation. The training process for this architecture is aided by photometric losses between the ground truth and novel renderings.

Point clouds, similar to triangular meshes, can be viewed within VR headsets and a comparison can be made between their method and ours. One of the downsides of point cloud is that each point in the scene needs to be represented individually, making it inefficient to scale to high-resolution data. Having efficient high-resolution results is important since it drastically improves the user’s experience with VR headsets. Because our approach relies on triangular meshes, we achieve better scalability compared to point clouds as a larger surface area can be effectively captured using only three points of a triangle, as opposed to employing individual points to depict the scene.

3.1.3 Neural radiance fields

Another popular option for view synthesis is with Radiance Fields. Recent studies in rendering radiance fields using deep neural networks - termed Neural Radiance Fields (NeRF) have allowed high-quality novel view synthesis from a collection of posed input images. In this research, Milendhall et al. represent static scenes as a 5D function that outputs color densities in each direction at each point in space. NeRFs are a fully-connected non-convolutional deep neural network that takes the 5D coordinate as an input to produce novel views[13]. This process goes as follows: first, camera rays are projected through a scene from all the input image angles, sampling 3D points and their corresponding colors at regular intervals; next, those points \((x, y, z)\) and their viewing directions \((\theta, \phi)\), which are extracted from the input image, are used by the NeRF to produce a set of color and densities; finally, those
colors and densities are accumulated into the novel 2D image by the help of volume rendering techniques.

A disadvantage of these methods is that they usually require more than one input view to generate novel views [15, 4]. Not to mention, the volumetric scene representation from NeRF [13] suffers from slow rendering time that is not suitable for running in common VR headsets. Our method addresses these research gaps by (1) operating on a single panorama input, which is easily obtained with even a smartphone, and (2) producing a highly efficient mesh representation that is compact enough to run in a VR headset.

3.1.4 CNNs for Multi-plane images

Another method to achieve view synthesis is through multi-plane images (MPIs), and these require a significantly fewer number of input images. MPIs are a series of parallel RGB images, each at different depths from a reference point, with an $\alpha$ channel to depict the visibility of each layer [29, 25, 6]. Rather than having variable depths for each pixel as shown in layered depth images, each RGB$\alpha$ plane has a collection of pixels with variable opacity, $\alpha$, and fixed depth. Zhou et al.’s method uses a fully convolutional neural network to extract the multi-plane image representation [29]. For each plane, the network predicts an alpha map and a blending weights image as well as a global background image. The blending weights image represents the percentage of the foreground and background sampled for each pixel at each layer. Zhou et al. conclude that color information in a scene can be extrapolated by just two images, a foreground and a background image, where the foreground image is usually the input image, and the background image is predicted by the network. Once the MPI is produced, novel views can be rendered by first applying a transformation to the RGB$\alpha$ layers per the target camera intrinsics followed by an alpha-composition of
the planes into the novel rendering. Both the planar transformation and the alpha composition are learnable parameters.

One limitation of this method is that the depth perception of objects is very dependent on the number of planes. The greater the disparity in artifacts of an image, the more planes are required to preserve realism. Our method produces a 3D mesh that is dependent on a depth map to maintain depth perception, however, each pixel has a depth value thus all artifacts of an image can be accurately represented. Additionally, our method stores the entire mesh on a single surface with layers rather than on multiple planes, making our method more storage efficient.

3.2 Panoramic View Synthesis

The following section outlines some more examples of view synthesis, particularly focusing on producing 360° content. The inputs to these examples vary, ranging from panoramic images like the ones we are using for our paper to multiple perspective images capturing different views.

3.2.1 Virtual Environment from Panoramic View

Audu et al’s 2013 paper on producing a virtual environment from stereo panoramic views is one of the earliest works on virtual environment production and set the precedent for later studies[3]. Their process involved passing a panoramic image through a view synthesis pipeline to generate a depth map. Then, they textured the depth map with the RGB data from the input image to produce a virtual environment. Despite their results not meeting expectations and their lack of full 6DoF, we can still draw inspiration from their approach.
Our baseline method for comparison is a textured mesh inspired by this method. Simple textured meshes expose gaps or a stretching effect upon translational movements, significantly affecting the user’s experience within the scene. In later sections, we will qualitatively compare the textured mesh with our method.

3.2.2 Multi-depth panoramas

Lin et al. propose a method to perform view synthesis by leveraging Multi-Depth panorama (MDP)[10]. MDPs consist of multiple RGBDα panoramas where the variable depth channel per plane gives detailed texturing to each layer. Their network works with $k$ number of perspective images representing different views using a special $360^\circ$ camera setup. A plane-sweep volume (PSV) is created from these images by first identifying the target view and rotating images from its four nearest cameras. A PSV consists of a collection of images reprojected to a target camera’s viewpoint[7]. The PSV is then processed by a 3D CNN to predict multi-plane images (MPIs) per perspective viewpoint which are then projected into the cylindrical coordinate system. These per-view MPIs are grouped by plane radius ranges and combined to produce per-view MDPs. The MDPs are blended to form a single global MDP that can be used to render novel views. To perform novel view synthesis from a global MDP, the predicted MDP is treated as a set of RGBα point clouds, similar to MPIs, and so they need to be projected onto the target image plane. Once projected, they alpha-composite the target image planes to produce the final novel view rendering.

Lin et al. use a multi-camera setup to produce a scene representation suitable for rendering novel views. However, these rigs are rare, expensive, and impractical for consumers. We choose to limit our input to a single image that can be created from consumer-grade hardware. Our method looks to render novel views of a scene using a single equirectangular panoramic image.
3.2.3 CNNs

In their study, Xu et al. investigate the utilization of a single panoramic image for novel view synthesis[28]. Their methodology involves the extraction of a dense feature map, a depth map, and a room layout from the source panorama with a Convolutional Neural Network (CNN). The dense feature map contains important elements of the images and it represents the contextual information of the scene. The room layout is composed of the edges and corners of a room based on the panoramic perspective and it represents the structural information of the scene. The depth map provides information about the spatial layout of objects within the scene. Following this extraction, the feature map and room layout are transformed to align with the target perspective through pixel-based mapping between the two viewpoints. They are then fused to synthesize the target panorama. The target panorama’s layout is extracted and compared with the transformed room layout from earlier to compute a layout consistency loss. The methodology is trained to minimize this loss. They train their model on 2 datasets: an easy set with minimal camera translation, consisting of 13,080 training images and 1,791 testing images, and a hard dataset with greater camera translations, consisting of 17,661 training images and 2,279 testing images. CNNs are currently the industry standard for feature extracting and this network architecture can be used to create properly detailed views, especially with larger viewpoint transformations.

One limitation of this method is that it does not produce a 3D environment; instead, it comprises a series of models that generate novel views based on camera location and translation. To function effectively in a VR setting, this model requires continuous processing of new views which is computationally inefficient. In contrast, our method generates a mesh with all pixels already generated and ready for VR use, enabling navigation without the need for extensive computation.
3.2.4 Multi-cylindrical images

In their research, Waidhofer et al. explore the use of consumer hardware images for rapidly capturing 360-degree scenes with free-viewpoint rendering in the region surrounding the point of capture[26]. This is achieved by employing multi-cylindrical images (MCIs). MCIs are created with a deep convolutional neural network similar to the network used to produce the MPIs mentioned earlier. Waidhofer et al.’s network utilizes a U-Net style design to produce semi-transparent cylindrical layers at varying depth levels. To accomplish view synthesis and produce a VR-ready environment from the MCIs, they create an environment consisting of a series of concentric cylindrical meshes, each corresponding to a layer in the MCI. The radius of these meshes is determined by their depths from the camera’s center and the heights of these cylinders are twice the inverse of their depth values. They are textured from the MCI layers by mapping each color value to the inside of each cylinder and alpha blending visually combines the layered meshes, resulting in a VR-ready environment with 6DoF.

This research aligns closely with ours. We also look to use images produced by consumer hardware for view synthesis, we also use a single panoramic input, and we also aim to perform real-time view synthesis for VR applications. There are two major downsides to their research, however: firstly, PanoSynthVR’s results are limited to 2048 × 1024 resolution panoramas; secondly, their method often results in a blurring effect from the blending of the pixels at different cylindrical layers. While this can give great metric scores, it comes at the cost of user experience. In contrast, our method not only works with up to 8K resolution panoramas but employs a mesh format that is not subject to the blurriness that MCIs can face.
Chapter 4

IMPLEMENTATION

Here we introduce our 3D Pano Inpainting pipeline, a method aimed at creating 360° scenes that can be viewed in a VR headset for an immersive experience. Our primary objective is to create a 3D mesh that surrounds the user’s field of view from all directions. Our secondary objective is to complement the 3D mesh by inpainted edges in occluded regions for photorealistic novel views that give the user a full 6 degrees of freedom. We aim for feasibility above all else and thus our method works with singular panoramic images, prioritizing practicality and accessibility for users without sacrificing their quality of experience.

We divide our pipeline into 4 different sections: (1) depth estimation, (2) inpainting, (3) projection, and (4) view synthesis. The key component of our research is based on the 3D photo inpainting framework introduced by Shih et al.[22]. Their methodology is designed to work with RGB-D images to produce a layered depth image (LDI) with inpainted color and depth of background pixels surrounding foreground objects. Figure 4.1 shows the 4 stages of our pipeline for panoramic view synthesis.

4.1 Input

Our 3D Pano Inpainting pipeline takes a single 360° equirectangular RGB image as an input and it can handle up to 8K resolution panoramas (7,680 × 3,840) on a V100 GPU. Because the depth estimation model can work on higher-resolution images, our pipeline’s bottleneck is GPU memory and VR headset RAM. Currently, the VR
Figure 4.1: With a single input panorama and a depth map generated by 360MonoDepth[19], our system uses a combination of inpainting and spherical projection to generate (1) a 3D mesh that is capable of VR rendering, and (2) novel panorama views [22].

headset RAM is limited to only work with 4K resolution panorama. During this, the headset functions at 60 Frames Per Second and consumes 1.351 GB of GPU Memory.

4.2 Depth Estimation

To create visually appealing environments for VR headsets, it is essential to utilize high-resolution RGB-D images. Utilizing higher-resolution images guarantees that the resulting 3D mesh offers the sharpest and most detailed view to the user. With this mesh clarity, we can differentiate small artifacts from their backgrounds and accurately represent the edges that need inpainting, enhancing the quality of intricacy of the virtual environment. Equally crucial is the generation of depth maps that ensure perfect alignment along their left and right edges. Such precision eases the stitching between the left and right edges of the produced mesh.

4.2.1 Boost-monodepth + wrap padding

In Shih et al.’s paper, they utilize boost-monodepth, a perspective depth estimation algorithm, to generate depth maps from 2D image inputs before proceeding
with the remainder of the inpainting methodology\cite{22}. However, simply using boostmonodepth on panoramic images leads to alignment adjustment issues in the depth map along the left and right edges. One approach to address this issue is through wrap padding, as discussed in the PanoSynthVR paper by Waidhofer et al.\cite{26}. This technique involves adding horizontal wrap padding to the input image. Specifically, the left half of the panorama is duplicated and concatenated to the right side, and vice versa. The idea is that the additional padding will help the depth estimation model use the connected version of the panorama to estimate depth. After running the padded image through boostmono-depth, the excess padding is cropped off, theoretically resolving any misalignments. However, despite these measures, we observed that the alignment issues along the edges remained unresolved. We attribute these distortions to boost-monodepth specifically, which we believe to be trained on perspective images where the subject is typically at the center of the image. As a result, it is less accurate at predicting depth as we move away from the center of an image, especially in the case of padded and equirectangular images.

4.2.2 360Monodepth

We adopt 360MonoDepth in our pipeline to address the challenge of accurately predicting spherical depth maps with aligned left and right edges. 360MonoDepth is specifically designed to estimate high-resolution depth maps from a single 360° monocular equirectangular input image. It can consistently produce high-resolution maps because it projects the 360° image into a series of perspective tangent images of size 2048 × 1024 pixels before running depth estimation. Then for each of these images, it independently predicts a depth map using the state-of-the-art in depth estimation, with the flexibility to swap out the depth estimation model to accommodate advancements in the field. The respective depths are reconfigured and aligned within
the spherical domain to correct any projection-related misalignment issues. These aligned disparity maps are ultimately blended to create a high-resolution spherical disparity map.

\[
disparity = \frac{1}{\text{depth}}
\]  

(4.1)

4.3 Building Meshes

With a high-resolution 360-degree photo and its corresponding depth map, we have a high-resolution RDB-D image, allowing us to leverage the 3D-photo inpainting framework. Shih et al.’s methodology utilize the mentioned input to generate a layered depth image (LDI) with inpainted color and depth [22]. The process involves several key steps: firstly, constructing a comprehensive LDI from an RGB-D image; then, identifying areas of discontinuity within the LDI; subsequently, employing a pre-trained inpainting model to synthesize the depth and color information for the background layer in these identified regions. The synthesis process takes into consideration the surrounding scene geometry and the visual characteristics of adjacent pixels. Finally, this synthesized region is integrated into the LDI, resulting in a more photorealistic 3D textured mesh. When adjusting the parameters for 3D photo inpainting, it is important to specify 'extrapolate borders' as 'False' and set 'thickness' to '0' as our coordinate projection formulas inherently align the mesh edges, eliminating the need for further extrapolation.

As shown in figure 4.2, the synthesis region covers a substantial portion of the occluded region, improving the overall visual appearance from the specific angle.
Figure 4.2: Example inpainting result from our system. (a) The black regions are gaps in the mesh caused by depth discontinuities. (b) The gaps have been filled with mesh geometry and inpainted.

4.4 Coordinate Projection

To create a fully immersive virtual reality environment, it is important to ensure that the mesh conforms to the perspective of the camera, wrapping around its view. Our initial step involves scaling the width and height coordinates of the LDI to fit within the ranges of $-\pi$ to $\pi$ and $-\frac{\pi}{2}$ to $\frac{\pi}{2}$, respectively. These scaled values correspond to our horizontal angle ($\theta$) and vertical angle ($\phi$). Then, we can project our mesh by calculating the Cartesian coordinates $(X,Y,Z)$ using the following formulas:

$$X = \sin(\theta)\cos(\phi) \quad (4.2)$$

$$Y = \sin(\phi) \quad (4.3)$$

$$Z = \cos(\theta)\cos(\phi) \quad (4.4)$$
Hence, our horizontal field of view is 360° around the camera, whereas the vertical field of view is 180°. Finally, we merge the two edges of our panoramic mesh by allowing the right edge points, represented by the final column of the LDI, to establish connections in a triangular mesh configuration to the left edge points, represented by the initial column.

4.5 View Synthesis

Once we have our mesh, it is easy to render different viewpoints as the inpainted edges eliminate obstructions from any potential movement. We consider both the camera’s pose and projection type to generate an image from the camera’s viewpoint. The camera’s pose is its position and orientation. This includes attributes such as the camera’s location and the direction it is facing. The camera’s projection type determines the type of image it is taking, for example, a perspective projection vs an equirectangular projection.
Chapter 5

EXPERIMENTAL SETUP

This chapter describes our experimentation setup to assess how well our model synthesizes novel views compared to a baseline model and the ground truth. We give a brief overview of our experimentation methodology:

1. We use a synthetic dataset that gives us equirectangular images and their inferred ground truth depth maps, which we will refer to as ‘viewpoint’ images. From the scenes we chose to reproduce in the dataset, We retrieved 5 random viewpoint panoramas. For each chosen viewpoint, we also extract 3 camera positions, their matrices, and their corresponding ground truth renderings.

2. We run the different mesh-producing frameworks under comparison on the viewpoint panorama, using the ground truth depth when needed. This process gives us a collection of 5 different mesh files per scene per method. All methods under comparison generate 3D meshes based on the viewpoint images and predict novel views from the different camera positions.

3. For each of the mesh files, we employ OpenGL to transform them into their cube map equivalent, allowing us to render spherical 360° images from the 3 distinct camera positions. These renderings are compared with their ground truth counterparts, specific to the camera positions, and given scores based on how close the mesh’s novel renderings are to the ground truth renderings.
5.1 Replica Dataset

We use 360 Replica Dataset Generator [2] to produce synthetic spherical datasets, comprised of several viewpoints. Here we use the Replica dataset[23] because it has been used in prior research papers to evaluate panoramic view synthesis models, such as MatryODShka. Each viewpoint contains three unique camera positions to capture diverse objects and scene geometries. In the following sections, we explore how MatryODShka renders the views for these camera positions.

5.1.1 Generate Viewpoints

MatryODShka’s dataset generator is designed to accurately simulate realistic indoor environments with few missing/incorrect geometry. This process starts with the development of custom equirectangular projections (ERP) and omnidirectional stereos (ODS) rendered for the Replica Dataset. They then sample random navigable floor positions in Replica. To get these navigable floor areas in the Replica dataset, they employ the use of the Facebook AI Habitat simulator[12]. The AI habitat is a simulation platform designed for research. It allows intelligent machines to train in a highly photorealistic 3D simulator and transfer the learned skills to the real world. Once a ground position is chosen, the next step is to determine the height at which the camera will be placed. They get the camera’s vertical position by sampling from a Gaussian distribution of human height. The randomly sampled perspective viewpoints from the AI habitat ensure that the scene is viewed from various viewpoints. At each of these positions, the left and right eye ODS images are rendered as well as an equirectangular image. However, our focus lies on just the equirectangular projections since that is what our pipeline uses. Each image has a dimension of 2880 pixels by 1440 pixels. For our evaluation, we base the creation of these datasets on
the layouts of "hotel 0," "office 0," and "room 0" from the Replica Dataset [23]. The layouts provide highly detailed reconstructions of indoor spaces, including features like glass, mirrors, and a variety of shapes.

5.1.2 Camera poses

To obtain camera poses (P) for each dataset, we save the camera-to-world transformation matrices from the dataset generator and perform a dot product of the target transformation matrix (T) with the inverse of the source transformation matrix (S).

\[ P = T \cdot S^{-1} \] (5.1)

We are given the source matrix as the sampled perspective viewpoint's camera-to-world transformation matrix. Our target matrices are also extracted from the dataset generator. The camera pose obtained from the formula above gives the unique camera position, displaying our novel views.

5.1.3 Ground Truth Depth

After removing the 1/16 scaling factor, we extract the depth matrices from the dataset generator and save them as the ground truth depth maps. These are necessary because our evaluation includes panoramic renderings of inpainted meshes generated from the ground truth depth maps, alongside panoramic renderings of inpainted meshes generated from 360MonoDepth depth maps [19]. For meshes processed with 360MonoDepth, we scale the input depth map by the median ratio between ground truth and input depth maps prior to generating the LDI. This adjustment ensures the preservation of the camera position within the viewpoint while minimizing errors.
5.2 Image Metrics

We evaluate the generated viewpoints from the following:

- 3D Pano Inpainting mesh with ground truth depth maps
- 3D Pano Inpainting mesh with 360MonoDepth generated depth map
- Textured mesh from the ground truth depth map

by comparing them to the ground truth viewpoints using three quantitative image similarity metrics: PSNR, SSIM, and LPIPS.

SSIM and PSNR are commonly used metrics for image comparison, assessing statistics derived directly from pixel values. PSNR stands for peak signal-to-noise ratio and this metric evaluates the difference in pixel values between two images using mean squared error (MSE). SSIM stands for structural similarity index and, unlike PSNR, it measures the similarities in properties, such as contrast and structure, between two images. LPIPS, a relatively recent metric, has gained popularity for measuring perceptual image patch similarity. This metric functions by comparing feature maps extracted using a deep convolutional neural network from a pair of images. These metrics prove useful in assessing the quality of novel views produced by the methods. Optimal results are defined by high PSNR values, low LPIPS values, and SSIM values nearing 1.
Table 6.1: Results of quantitative evaluation on the synthetic scene "hotel 0". Values are the averages of the three renderings per viewpoint. The best value of each metric is shown in bold.

<table>
<thead>
<tr>
<th>Type Method</th>
<th>PSNR↑</th>
<th>SSIM↑</th>
<th>LPIPS↓</th>
</tr>
</thead>
<tbody>
<tr>
<td>3D Pano Inpainting</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Hotel 0 Viewpoint 0</td>
<td>25.59</td>
<td>0.958</td>
<td>4.243e-06</td>
</tr>
<tr>
<td>Hotel 0 Viewpoint 1</td>
<td>25.68</td>
<td>0.957</td>
<td>3.957e-06</td>
</tr>
<tr>
<td>Hotel 0 Viewpoint 2</td>
<td>22.11</td>
<td>0.948</td>
<td>1.940e-06</td>
</tr>
<tr>
<td>Hotel 0 Viewpoint 3</td>
<td>24.49</td>
<td>0.962</td>
<td>5.770e-06</td>
</tr>
<tr>
<td>Hotel 0 Viewpoint 4</td>
<td>22.93</td>
<td>0.960</td>
<td>9.156e-06</td>
</tr>
<tr>
<td>3D Pano Inpainting w/ Monodepth</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Hotel 0 Viewpoint 0</td>
<td>21.74</td>
<td>0.871</td>
<td>9.098e-06</td>
</tr>
<tr>
<td>Hotel 0 Viewpoint 1</td>
<td>21.99</td>
<td>0.874</td>
<td>6.590e-06</td>
</tr>
<tr>
<td>Hotel 0 Viewpoint 2</td>
<td>21.58</td>
<td>0.883</td>
<td>9.186e-06</td>
</tr>
<tr>
<td>Hotel 0 Viewpoint 3</td>
<td>19.22</td>
<td>0.874</td>
<td>1.287e-06</td>
</tr>
<tr>
<td>Hotel 0 Viewpoint 4</td>
<td>20.23</td>
<td>0.897</td>
<td>1.271e-06</td>
</tr>
<tr>
<td>Textured Mesh</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Hotel 0 Viewpoint 0</td>
<td>25.41</td>
<td>0.963</td>
<td>4.113e-06</td>
</tr>
<tr>
<td>Hotel 0 Viewpoint 1</td>
<td>25.49</td>
<td>0.959</td>
<td>3.733e-06</td>
</tr>
<tr>
<td>Hotel 0 Viewpoint 2</td>
<td>25.17</td>
<td>0.961</td>
<td>4.651e-06</td>
</tr>
<tr>
<td>Hotel 0 Viewpoint 3</td>
<td>25.27</td>
<td>0.969</td>
<td>4.176e-06</td>
</tr>
<tr>
<td>Hotel 0 Viewpoint 4</td>
<td>23.36</td>
<td>0.965</td>
<td>6.979e-06</td>
</tr>
</tbody>
</table>

6.1 Quantitative Evaluation

With the new camera pose transformation and the synthetic equirectangular panoramic images, we are able to generate a viewpoint with the camera position shifted from the original position. This shift unveils previously obscured features because of inpainting, particularly in areas not visible from the original position. Transformation matrices for each face of a cubemap (F, L, B, R, D, U) are defined using rotations
Table 6.2: Results of quantitative evaluation on the synthetic scene “office 0”. Values are the averages of the three renderings per viewpoint. The best value of each metric is shown in bold.

<table>
<thead>
<tr>
<th>Type Method</th>
<th>PSNR↑</th>
<th>SSIM↑</th>
<th>LPIPS↓</th>
</tr>
</thead>
<tbody>
<tr>
<td>3D Pano Inpainting</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Office 0 Viewpoint 0</td>
<td>25.37</td>
<td>0.956</td>
<td>5.328e-06</td>
</tr>
<tr>
<td>Office 0 Viewpoint 1</td>
<td>25.20</td>
<td>0.967</td>
<td>4.674e-06</td>
</tr>
<tr>
<td>Office 0 Viewpoint 2</td>
<td>25.04</td>
<td>0.968</td>
<td>4.002e-06</td>
</tr>
<tr>
<td>Office 0 Viewpoint 3</td>
<td>24.83</td>
<td>0.964</td>
<td>5.376e-06</td>
</tr>
<tr>
<td>Office 0 Viewpoint 4</td>
<td>25.05</td>
<td>0.956</td>
<td>5.425e-06</td>
</tr>
<tr>
<td>3D Pano Inpainting w/ Monodepth</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Office 0 Viewpoint 0</td>
<td>22.69</td>
<td>0.841</td>
<td>6.545e-06</td>
</tr>
<tr>
<td>Office 0 Viewpoint 1</td>
<td>23.40</td>
<td>0.889</td>
<td>5.457e-06</td>
</tr>
<tr>
<td>Office 0 Viewpoint 2</td>
<td>23.44</td>
<td>0.888</td>
<td>5.125e-06</td>
</tr>
<tr>
<td>Office 0 Viewpoint 3</td>
<td>22.43</td>
<td>0.853</td>
<td>7.848e-06</td>
</tr>
<tr>
<td>Office 0 Viewpoint 4</td>
<td>23.73</td>
<td>0.848</td>
<td>5.875e-06</td>
</tr>
<tr>
<td>Textured Mesh</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Office 0 Viewpoint 0</td>
<td>26.24</td>
<td>0.959</td>
<td>3.582e-06</td>
</tr>
<tr>
<td>Office 0 Viewpoint 1</td>
<td>25.21</td>
<td>0.968</td>
<td>4.434e-06</td>
</tr>
<tr>
<td>Office 0 Viewpoint 2</td>
<td>24.99</td>
<td>0.971</td>
<td>4.178e-06</td>
</tr>
<tr>
<td>Office 0 Viewpoint 3</td>
<td>25.39</td>
<td>0.964</td>
<td>3.916e-06</td>
</tr>
<tr>
<td>Office 0 Viewpoint 4</td>
<td>26.76</td>
<td>0.958</td>
<td>3.207e-06</td>
</tr>
</tbody>
</table>
of \( \frac{\pi}{2} \) radians. These transformations are applied to the mesh scene and we extract the color and depth matrices. The cubemap is converted to an equirectangular image using the py360convert library [24]. The equirectangular images are the renderings for our mesh scenes, and can now be compared with the ground truth to derive the evaluation metrics.

Tables 6.1, 6.2, and 6.3 contain the metrics for our evaluation. The textured mesh representation performs the best across all datasets, likely due to its stretching effect which preserves color in renderings. In contrast, the inpainted mesh on the ground truth depth map did not fully inpaint the area behind objects, exposing gaps. This is likely because the edge inpainting model fails to recognize the potential for further edge production, thereby hindering the color and depth inpainting process in those

Table 6.3: Results of quantitative evaluation on the synthetic scene ”room 0”. Values are the averages of the three renderings per viewpoint. The best value of each metric is shown in bold.
regions. However, it’s worth noting that the textured mesh representation only holds a marginal advantage over the ground truth depth 3D Pano Inpainting method in most viewpoints.

Despite presenting the worst average metrics, the 3D Pano Inpainting method with 360MonoDepth stands as our most visually effective approach, which is discussed in the qualitative evaluation and discussion sections. The worse metrics stem from a disparity in depth map scaling compared to the ground truth, resulting in a slightly different camera position than the ground truth. Consequently, this discrepancy offsets all pixel values, leading to substantial penalties in PSNR, SSIM, and LPIPS.

6.2 Qualitative Evaluation

While the quantitative evaluation methods indicate that the textured mesh representation on the ground truth depth has better metrics than the other methods, we found that the inpainting mesh consistently outperformed the textured mesh in terms of visual quality and overall rendering.

Figure 6.1 shows the novel view rendering from "room 0" using three different methods. The ground truth rendering is compared with the renderings from our 3D Pano Inpainting pipeline using the ground truth depth map and from a textured mesh representation using a ground truth map, When we zoom into the artifacts from the 3 methods, we notice that textured mesh representation has the most visible and obvious distortion, whereas the 360MonoDepth with inpainting has the least.

Figure 6.2 provides confirmation that the 3D Pano Inpainting with 360MonoDepth method effectively addresses the issue of stretched pixels, leading to sharper edges, which would enhance the overall experience in a VR headset. The sharper edge also
uncovers information from behind the object that was previously obscured by the artifact, further improving the perceptual quality.

6.3 Discussion

The textured mesh has noticeable texture “stretching” artifacts near the edges of objects. Since the artifact still preserves the color of the surrounding pixels, this boosts the quantitative performance but results in an unnatural and low-quality visual appearance.

On the other hand, the inpainted meshes showcase better performance around object edges, presenting well-defined boundaries. However, the 3D Pano Inpainting method without 360MonoDepth leaves gaps near some of these edges. Which we believe to be caused by the dependence on the edge detection model predicting edges to proceed with extending the colors and depths. 3D-photo inpainting explains that the framework keeps generating pixels as long as new edges are detected[22].

In contrast, the 3D Pano Inpainting mesh with 360MonoDepth achieves the most appealing visual results, devoid of stretching artifacts and gaps. The imperfection of 360Monodepth in predicting depth maps proved advantageous in this case as the pipeline managed to cover more holes in the mesh compared to the ground truth version. Unfortunately, this method lags behind quantitatively due to a persistent issue with depth map scaling in 360MonoDepth.
Figure 6.1: Qualitative comparisons for room 0
Figure 6.2: Comparison of a textured mesh model (inset bottom) and our Panolpainting with 360MonoDepth (inset top).

Figure 6.3: Illustration of the conversion between a cubemap and an equirectangular projection. [24]
Chapter 7

FUTURE WORKS

In this chapter, we look at potential avenues for future research which can be explored based on the results produced by this paper. Although 3D Pano Inpainting proves to be an effective method in panoramic view synthesis, there exist many opportunities within this project to further refine and improve upon the methods investigated here. Additionally, research in view synthesis is constantly evolving, presenting areas of innovative techniques that can be explored for VR applications.

7.1 Depth Scaling Discrepancies

Our evaluation method yielded subpar results. Although the 3D Pano Inpainting with 360Monodepth showcased great qualitative results, quantitative metrics revealed a different perspective. We noticed a misalignment in the viewing direction between ground truths and renderings from 3D Pano Inpainting + 360Monodepth. This slight misalignment significantly affected the accuracy of the evaluation metrics, resulting in an incorrect numerical assessment. We believe that this discrepancy arises because the ground truth depth map and the 360Monodepth depth map are incorrectly scaled. Our initial approach was just to multiply each depth value in the predicted map by the median ratio between ground truth and input depth maps. Fixing the scaling is an area of expansion for this project.

We believe that the offset scaling can be treated as a regression problem. Given that 360Monodepth is an estimation tool, it is safe to assume that it will produce some outliers during depth approximation and thus using a model that is less sensitive to
outliers would produce better results. Since we are working with depth maps and our scaling was only slightly off, we can assume that there is at least some linearity between the ground truth and estimated depths. Because we want to scale the estimated depth to the ground truth scale, our independent variable is the 360Monodepth depth and our dependent variable is the ground truth depth. Once the linear model fits the data, we can use it to transform the estimated depth to the ground truth scale.

7.2 Comparison with View Synthesis Models

During the evaluation process, our approach involved a comparison between 3D Pano Inpainting and a baseline textured mesh derived from the ground truth depth. An extension to our evaluation process is to compare this paper’s results with other panoramic view synthesis models. Two such models that could be potential candidates are PanoSynthVR [26] and MatryODShka [2].

PanoSynthVR is relevant because it is conceptually close to our research[26]. It leverages Multi-cylinder images to create a mesh that surrounds the user’s field of view, providing a full 6 degrees of freedom. This model involves using a single cylindrical panorama to create an immersive VR environment. One challenge with this is that their method involves cylindrical projections and ours involves spherical projections. Therefore, a preliminary step involving generating cylindrical projection images of the same dataset will have to take place.

Alternatively, MatryODShka uses Omni-Directional-Stereo projection images. Their paper, however, would be the easiest to evaluate against our method because they use the same synthetic dataset for assessing their model’s performance. A shared dataset could simplify the results’ analysis process.
7.3 Edge Inpainting

When employing the 3D Pano Inpainting pipeline with the ground truth depth, we encountered gaps around certain edges. This can be another area of future improvement for this project because we should look to fill as many of those gaps as possible. The inpainting framework’s ability to continuously render new pixels is dependent on whether new edges are being produced in synthesis regions. Thus, if the edge inpainting network ceases to generate new edges, so do the color and depth inpainting networks. This reliance is necessary to give structure to synthesis regions before rendering new pixels.

A naïve way to ensure that the process continues to produce pixels is by re-running the inpainting process. The current methodology already runs two inpainting cycles, so theoretically a third time should help cover more of that area. In addition to this, a better edge detection model can drastically help move produce more edges and better meshes for control.

7.4 UV Mapping

For each of the vertices in a polygon mesh, the positional coordinates as well as RGB color and depth values are stored. Additionally, inpainted regions add another layer of complexity which leads to slower rendering times. These regions demand more resources during the rendering process.

We believe that UV mapping can help reduce the rendering time for inpainted meshes. This idea is that these regions will unwrapped to the UV coordinate system, allowing for textures to be pre-computed and bypassing the need for allocating computational resources during the rendering process.
In this paper, we introduce 3D Pano Inpainting, a framework for building an immersive 360-degree scene for Virtual Reality headsets with only a single input image. The 3D Pano Inpainting pipeline leverages a depth estimating model and an inpainting framework, coupled with some basic geometric pixel manipulation in order to transform a single equirectangular RGB panorama into a dynamic, three-dimensional environment with full 6 degrees of freedom and motion parallax support.

8.1 Key Findings

We find that despite our methods marginally falling short against the textured mesh quantitatively, qualitatively, our pipeline proves far superior. The key distinction in our method’s superiority lies primarily in its minimal occurrence of stretching artifacts, particularly noticeable with head movement and especially when viewed in a VR headset.

8.2 Contributions

The main contributions of our work are:

- We successfully create a pipeline that generates a mesh capable of supporting 360-degree view synthesis with motion parallax from a single equirectangular input panorama.
• We evaluate our results by analyzing the quality of images from rendered views of different meshes using a synthetic dataset. Our evaluation incorporates image quality metrics such as PSNR, SSIM, and LPIPS.

8.3 Practical Implications

The practical implications of our research are vast. Ultimately we look to use this method in the development of lightweight VR applications for consumer use. By allowing scenes to be recreated with just a singular panorama, capable of being captured with available equipment, we push for an accessible approach to virtual reality. Thus, 3D Pano Inpainting is most applicable for real estate, virtual tourism, and gaming.

Within real estate and virtual tourism, our method offers the potential for individuals to use their smartphones and create a virtual representation of a property listing or tourist attraction so they can gain an understanding of the space without physically being there. Our method can also help the gaming industry by allowing creators to recreate realistic scenes from real-world environments and embed them within their games. This method can significantly help game developers produce a seemingly realistic environment without having to buy special equipment for it.

8.4 Limitations

The 3D Pano Inpainting pipeline is limited in the following ways: (1) polygon meshes are very complex, especially in inpainted areas, thus it takes much longer to render views; (2) Quantitatively, this method lags against simple textured meshes.
8.5 Closing Remarks

With the advancements in machine learning research, there is no doubt that new methods for view synthesis will continue to evolve and improve. With that, we can expect these methods to be adapted for singular panoramic inputs and further refined to enhance the immersive experience of virtual environments. This thesis marks a step in the direction of practical and accessible VR and we hope that this paper will inspire further breakthroughs in the field of panoramic view synthesis.
BIBLIOGRAPHY


conference on computer vision and pattern recognition, pages 5515–5524, 2016.


synthesis from a single indoor panorama. In Proceedings of the IEEE/CVF
Conference on Computer Vision and Pattern Recognition, pages


appearance flow. In Computer Vision–ECCV 2016: 14th European
Conference, Amsterdam, The Netherlands, October 11–14, 2016,