Mono2VR: Exploring Immersive Experiences of Monocular Videos

Anthony Dickson [1], Wei Hong Lo [1], Felix Schrimper[1], Alistair Knott [2], Jonathan Ventura [3], Stefanie Zollmann [1,4*]

[1*]School of Computing, University of Otago, New Zealand.
[2]School of Engineering and Computer Science,, University of Wellington, New Zealand.
[3]Department of Computer Science & Software Engineering, California Polytechnic State University, USA.
[4]Department of Computer Science, Aarhus University, Denmark.

*Corresponding author(s). E-mail(s): stefanie.zollann@otago.ac.nz;

Abstract

Despite the growing popularity of VR headsets, most personal videos remain limited to flat 2D displays and lack the depth and motion cues needed for immersive playback. Converting monocular videos into 3D experiences suitable for VR remains a challenge due to the complexity and computational demands of existing solutions.

We present Mono2VR, a system that transforms standard monocular videos into immersive 3D content for VR headsets, with minimal processing time and modest hardware requirements. Unlike recent high-fidelity methods that are impractical for longer videos or real-time use, Mono2VR runs on consumer hardware in minutes per second of video.

Our pipeline estimates camera parameters and depth maps to reconstruct both dynamic foreground and static background elements. The resulting 3D videos support stereoscopic playback and head-motion parallax, enhancing immersion.

We evaluated Mono2VR both technically and in a user study, where participants rated our output on par with ground truth 3D content. These results highlight Mono2VR's potential to make immersive video experiences accessible to a broad audience.

Keywords: VR, Immersive Experience, View Synthesis

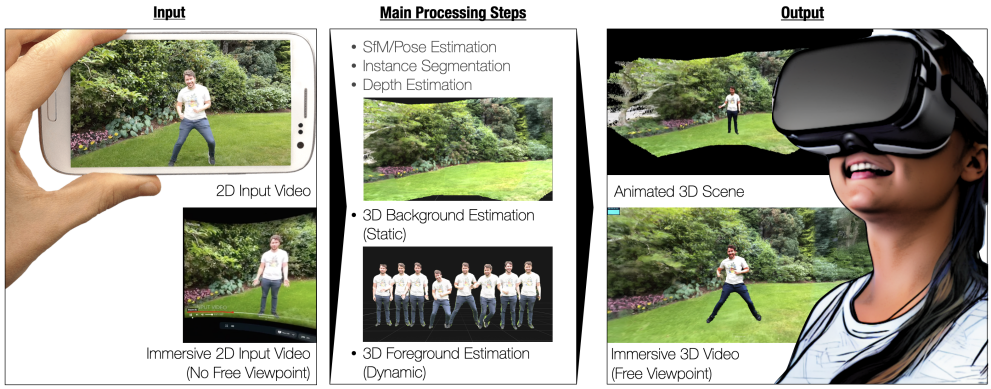


Fig. 1: Using Mono2VR to create and experience a 3D video in VR. Currently, 2D input video can only be watched on a virtual screen in a VR headset. This option does not allow for a free viewpoint selection and creates a cinema-like experience (Bottom Left). Instead, Mono2VR converts an input video into an immersive 3D video by 1) creating a 3D representation of the dynamic foreground for each frame of the video sequence, 2) creating a 3D representation of the static background, 3) and rendering the 3D video in a VR headset at interactive frame rates. The user can then mostly freely explore the immersive 3D video in VR

1 Introduction

Virtual Reality (VR) headsets have become increasingly affordable and accessible, leading to a growing interest in immersive media experiences beyond gaming. One recent example is Apple's introduction of spatial videos, which were launched alongside the Apple Vision Pro headset[1]. This demonstrates the potential for everyday users to engage with immersive personal content. However, creating compelling VR video experiences remains technically challenging. Most available 360-degree videos are monoscopic, which means they lack depth information and do not support motion parallax. As a result, they provide limited immersion and are poorly suited for VR playback. While stereoscopic 360-degree capture can improve depth perception, it requires specialized camera rigs and complex workflows [1–3], making it inaccessible to most everyday content creators.

Recent advances in image-based rendering (IBR) provide alternatives by enabling depth-aware novel view synthesis from monocular video. Techniques such as layered representations [4], light fields [5], Neural Radiance Fields (NeRF) [6], and 3D Gaussian Splatting [7] can reconstruct geometry and appearance from sparse viewpoints. While promising for offline applications, many of these methods are computationally intensive. For example, Neural Scene Flow Fields (NSFF) [8] requires two days to process a one-second video clip using two NVIDIA V100 GPUs and an additional 6 seconds per 512×288 frame for rendering, making it unsuitable for interactive VR scenarios. Although 4D Gaussian Splatting [9] significantly improves rendering speed and achieves approximately 14 frames per second, it still requires minutes of preprocessing per video and does not support real-time interactivity in head-mounted displays.

[1] https://support.apple.com/en-nz/guide/apple-vision-pro/dev7068c3c93/visionos

In addition to performance limitations, many of these methods do not compensate for the egomotion of the recording camera. This limitation can result in a mismatch between visual motion and head movement during VR playback, leading to discomfort or motion sickness [10]. Methods based on monocular depth estimation or multi-plane image (MPI) representations [5] also suffer from this issue and often lack sufficient depth fidelity for immersive parallax effects.

These limitations highlight the need for VR video playback solutions that support egomotion compensation, deliver motion parallax, and run at interactive frame rates with minimal preprocessing.

	Standard 2D Video on Screen in VR	RGB-D videos	MPI video	Neural Scene Flow Fields	4D Gaussian Splatting	Mono2VR (Ours)
Compensate Egomotion	No	No	No	Yes	Yes	Yes
Motion Parallax	No	Yes	Yes	Yes	Yes	Yes
Interactive Rendering in VR	Yes	Yes	Yes	No	No	Yes
Processing times	No processing required	Minutes	Minutes	Days	Minutes	Minutes

Table 1: Comparison of current methods for 3D videos for their feasibility for VR replays. Egomotion of the videos has been shown to be a factor for motion sickness [11]. Motion parallax is an important depth cue, and a lack of motion parallax can create motion sickness [10]. Interactive framerates are important for an immersive VR experience. Furthermore, processing times are an important factor to consider; while content creators are accustomed to a couple of minutes of processing for video editing and transcoding, multiple days of processing time is a limiting factor.

In this work, we describe Mono2VR, an end-to-end pipeline for creating and viewing immersive 3D videos in VR from monocular video[2]. Mono2VR combines traditional computer vision techniques with state-of-the-art deep learning models to achieve a unique approach to VR-capable 3D video reconstruction. In contrast to recent work on NeRF and GS-based methods, the main focus of our work is to achieve high framerates and interactive rendering on VR headsets. Furthermore, Mono2VR has substantially lower hardware requirements and processing times.

To evaluate the capabilities of Mono2VR, we conducted comprehensive technical evaluations and a user study in VR. Our evaluation covers a range of aspects, including rendering performance and file sizes for VR headsets, accuracy in camera trajectory estimation, mesh fusion based on depth data, processing time, and VRAM usage. Moreover, we present the first user study conducted for 3D videos generated using this approach, gathering valuable insights on user experience, presence, depth perception, and visual quality—an important aspect that is often ignored in other work. We make the source code for running Mono2VR and the experiments in this paper publicly available[3], contributing to the accessibility and reproducibility of this research.

[2]We note that Mono2VR appeared under the name 'VRVideos' in a work-in-progress publication of ours [12] and 'HIVE' in the code repository. These names all refer to the same method.

[3]https://github.com/AnthonyDickson/HIVE

2 Related Work

Extracting 3D data of static scenes from multiple photographs using structure-from-motion (SfM) methods is a well-established field [13]. Recent advancements focused on methods to create immersive 3D photographs that enable novel view synthesis with motion parallax. However, the extraction of 3D data from monocular videos is still challenging.

2.1 3D Photographs

The extraction of 3D information from a single photograph has evolved from *Tour Into the Picture* [14], one of the first methods that generate a simple 3D model from user-defined planes over more automatic methods for creating popup-like 3D representations [15] to sophisticated probabilistic depth estimations [16]. [4, 17, 18] leverage learned models for depth estimation and inpainting to produce realistic view synthesis from a single image. Hedman et al.'s [19] works to construct 3D panoramas with realistic motion parallax from a single mobile device with dual cameras, and also works with monocular video and estimated depth data. Mildenhall et al. proposed an approach for view synthesis based on an unstructured grid of input views from a video [5]. Their approach renders novel views by blending local light fields that are represented by Multi-Plane Image (MPI)s. MPIs were also used to synthesize novel views of a scene from an input photograph in the work by Tucker et al. [20]. However, there is only limited work on how well these methods work for content creation for VR experiences. While Dickson et al. [21] investigated the benchmarking of monocular depth estimation methods in the context of content creation for VR and Waidhofer et al. [22] investigated the impact of different depth representations of 360 images on presence and perceived quality, there is still limited knowledge on the feasibility of these content creation methods for VR experiences.

2.2 3D Video

While the above methods target static scenes, methods targeting dynamic scenes have been developing rapidly. Wang et al. [23] use a neural network trained to synthesize novel views from layered depth images and 2D image features, estimating appearance changes between a pair of photos. However, their approach is limited to two frames and takes 0.71s to render each frame ($\sim$1 FPS), and is therefore unsuitable for interactive applications and processing videos. NeRF-based methods [6, 24–26] can also produce photo-realistic novel-viewpoints of static scenes from a relatively small set of images. The implicit representation inherent to neural networks helps address many of the challenges in reconstructing dynamic 3D scenes with estimated depth and Truncated Signed Distance Function (TSDF) fusion-based approaches [27]. Initially, NeRFs had long training times and low framerates, but recent developments have seen the training time and render frame times significantly reduced [28], making them more suitable for interactive applications.

NeRFs have also been applied to dynamic video scenes [8, 29–33]. However, the efficiency gains have not carried over to dynamic video NeRF methods. NeRF methods often have long compute times in the order of hours (and sometimes days) long render times in the order of seconds per frame, and require expensive GPU hardware [8, 33]. The low framerates of these methods limit their applicability for interactive applications such as VR. Fang et al. [34] used an explicit representation to reduce training time but at the cost of lower quality view synthesis.

Their approach reduces training time down to minutes, however there is no mention of render framerates and still requires expensive GPU hardware.

3D Gaussian Splatting (3D-GS) [7] has many of the advantages of methods utilizing implicit and explicit representations. 3D-GS methods produce high quality novel viewpoint rendering of static scenes, often better than NeRF methods, while training in minutes instead of hours and rendering at high framerates (160-180 fps at 1080p). These improvements come from the use of 3D Gaussians which enables computation on empty space to be skipped and rasterization-based rendering. Initially, 3D-GS methods had high hardware requirements with the reference implementation of [7] requiring 24 GB of VRAM. Recent methods such as Wu et al.'s work [9] have significantly lower hardware requirements, largely eliminating this issue. However, the main challenge in using the approaches that reconstruct dynamic scenes from monocular video (e.g., [9, 35]) is that they do not achieve interactive framerates required for VR rendering (e.g. Wu et al: 14 FPS [9]). So while NeRF and 3D-GS methods achieve high-quality results for monocular video, their sub-interactive framerates make them unsuitable for interactive applications, especially VR.

Earlier methods have also been proposed to extract 3D representations from videos. However, they are either trained for a specific use case such as soccer [36], require a significant amount of user input [37], focus on static or rotating cameras [38] or focus on extracting dynamic content only for Mixed Reality applications [39] that do not need the background reconstructed. Dickson et al. [12] presented early work on a VR video representation that is suitable for VR rendering but without further investigation on feasibility.

2.3 360 VR Videos

There is more research on 360 VR videos, such as the work by Serrano et al. [10]. In their work, Serrano et al. propose a method that creates a layered representation of the 360 input video and use this for rendering in VR. Within their work, they investigate user preference and motion sickness compared to a conventional 360 (3 degrees of freedom) video. Their layered representation reduces motion sickness and scores better on user preferences. However, this method can only be used for 360 input videos. Thus there is still a lack of methods that are suitable for standard videos and limited knowledge on how well these methods work for creating VR experiences.

2.4 Summary

In our work, we combine traditional geometric methods with machine learning-based depth estimation and a layered representation to capture dynamic contents to reduce the computation times and complexity of the VR videos to make them suitable for replay in a VR headset.

In summary, the options for experiencing casually captured videos in VR are still limited. The most popular option currently is to watch videos on a virtual "cinema" screen as, for instance, offered by YouTube. Additionally, RGB-D videos and MPI videos are an option and can be computed quickly from standard 2D videos. Both offer some parallax, although this is more limited for MPI videos as the viewer cannot move far away from the position from where the video was captured. However, both RGB-D videos and MPI videos have a major disadvantage that they do not compensate for the egomotion of the capturing camera (Table 1) which can cause motion sickness [11].

3 Mono2VR System

For creating and rendering VR videos based on monocular 2D video input, we created the Mono2VR system. Mono2VR consists of four main steps: (1) data preparation; (2) creating a 3D representation of the static background; (3) creating a 3D triangle mesh representation of the dynamic foreground for each frame of the video sequence; and (4) rendering the 3D video (Figure 1). Our method takes monocular video as input but is also flexible enough to support RGB-D datasets (such as the TUM data set [40] or RGB-D datasets captured on an iPhone[4]).

3.1 Data Preparation

In the first step, we estimate depth data, dynamic foreground object segmentation masks, and the camera parameters for each frame within the video sequence. This information is vital for subsequent computations that involve the static background and the dynamic foreground.

3.1.1 Depth Map Estimation

For depth map estimation, we use the DPT depth estimation model [41] on each of the RGB frames of the input video. We use the model weights that were fine-tuned on the NYU dataset [42] and produce depth values that roughly correspond to meters (the model outputs depth in the interval [0, 10]). We store a depth map for each video input frame for further processing.

3.1.2 Segmentation Masks

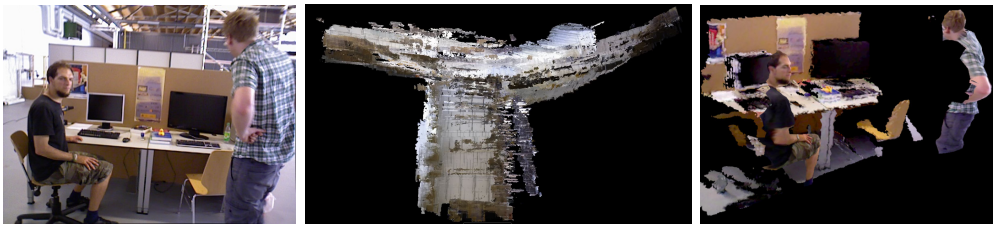
For the segmentation of the dynamic foreground, we compute instance segmentation masks for each frame of the video and derive binary masks for regions that are likely to contain dynamic foreground objects in the video. In our implementation, we limited the detection to people, as they are the most likely dynamic elements in our scenario. However, the instance segmentation can flexibly be extended to include any other dynamic element, such as cars, trains, or animals. We use Detectron2 for our implementation [43]. As a result, we get a segmentation mask for each video frame.

3.1.3 Camera Pose Estimation

We use COLMAP [44] to estimate the camera intrinsic and extrinsic parameters. Since running COLMAP on many frames (e.g., several hundred) can take several hours, we use a 'frame step' parameter to sample a subset of frames to lower the processing time and interpolate the missing poses. COLMAP works best with static scenes, so we use the segmentation masks from the previous step to exclude dynamic regions in the video to improve the estimated camera parameters.

COLMAP pose data is subject to an unknown scale factor. This can lead to misalignment in the 3D reconstruction due to discrepancies in the scale of the depth maps and translation vectors of COLMAP poses (Figure 2). We calculate a scaling factor to harmonize the pose and depth data scale. The scaling factor, denoted by σ, is derived from the median ratio of nonzero

[4] https://github.com/strayrobots/scanner



(a) Sample Frame (b) Misaligned Frames (c) Aligned Mesh

Fig. 2: When the scale of the depth maps and the COLMAP poses do not match it can lead to poor frame alignment (Figure 2b). Figure 2a shows one of the input frames and Figure 2c shows the resulting mesh when the depth and pose scales are aligned

depth values in the dataset to nonzero depth values from COLMAP:

$$\sigma = \text{median}(\frac{d_{\text{dataset}}}{d_{\text{COLMAP}}}). \tag{1}$$

We use the depth values from all frames, but only from pixels that are nonzero in both depth map sources. The depth values are divided element by element, and the median value is calculated from the result.

This approach is inspired by Tucker and Snavely [20]. Similar to the work of Waidhofer et al.[22], we use the median instead of the mean to make the scaling factor more robust to outliers in the COLMAP depth maps. The scaled COLMAP poses are then defined as:

$$\mathbf{R}_{\text{scaled}} = \mathbf{R}_{\text{COLMAP}} \tag{2}$$

$$\mathbf{t}_{\text{scaled}} = \sigma \mathbf{t}_{\text{COLMAP}} \tag{3}$$

where R denotes a rotation matrix and t a translation vector.

The scaling factor on its own only matches the scales of two depth map sources and does not guarantee that pose data scaled with this scale factor will match, or be close to, real-world measurements. It is only in conjunction with metric-scale depth maps that this approach can recover metric-scale pose data from COLMAP. The requirement for metric-scale depth data is, in part, what motivated our choice in depth estimation model.

3.2 Static Background Mesh Reconstruction

Reconstructing the static background of the video is a multi-step process. We adapt TSDF Fusion [45], which is intended for combining pre-registered RGB and *sensor* depth video data into a TSDF volume. In contrast to sensor data used in the original algorithm, our estimated depth data is not geometrically consistent. To adjust for this inconsistency, we add pre-processing steps to minimize visual artifacts from overlapping frames.

Simply using all frames, or uniformly sampling frames, often leads to blurry texture details due to many overlapping frames and small errors in the estimated pose data. We add a frame

sampling step that chooses the set of frames that mutually overlap no more than a given percentage threshold. We first uniformly sample one frame for every 30 frames to remove redundant frames that are not needed for background reconstruction. We then add the first frame as the first 'key frame'. Then we iterate through the remaining frames and test for 'overlap' by projecting points in the current frame onto the other key frames. We reject a frame if the points visible from a given key frame cover an area greater than the specified percentage threshold, i.e. we reject a frame if it does not add enough new information. Otherwise, we add the frame to the set of key frames and move onto the next frame. This frame sampling step greatly reduces the number of frames considered for background reconstruction and reduces blur artifacts.

To exclude dynamic elements from the background mesh, we apply the results of our instance segmentation. Contrary to Newcombe et al. [46], which estimates voxel representation for the dynamic elements of a scene, we focus on the static background rather than the dynamic elements. We dilate instance segmentation masks to create an error margin, preventing dynamic foreground pixels from entering the background mesh. To fill the "holes" behind the dynamic foreground elements, we apply LaMa inpainting [47] for the RGB images and the Fast Marching Method inpainting method [48] for the depth maps to preserve the precision of the depth values.

Each video frame is then integrated into the voxel volume observations using TSDF fusion. The voxel representation incorporates new observations from the RGB image, masked depth map, and camera pose per frame, all through a weighted average. The voxel's signed distance field (SDF) and color are updated, adjusting the SDF value if the voxel is occupied or setting it if unoccupied. The voxel's color and distance are updated by blending the previous and new observations based on the observation weight and the cumulative weights volume. The observation weight w_O is, thereby, a scalar value used when adding a new observation (RGB-D frame) to control the influence of the new observation on the existing TSDF values (w_O set to 1.0 in our implementation).

After integrating all frames, we generate a mesh using the marching cubes algorithm [49]. The final step we perform is gamma correction for the vertex colors[5]. The mesh is saved as a textured triangle mesh file and compressed to reduce file sizes.

3.3 Dynamic Mesh Reconstruction

The next step is to compute a mesh representation for parts of the scene that change each frame. Our approach is based on the meshing technique from the paper "Soccer on Your Tabletop" [36] and creates a textured triangle mesh from an RGB frame, depth map, camera matrix, camera pose, and instance segmentation mask.

We create a 3D point cloud by projecting the 2.5D points (2D image coordinates + depth) into 3D world coordinates by applying the depth, inverse camera matrix and inverse pose T_{inv}

$$\mathbf{T}_{\text{inv}} = \left[\mathbf{R}^T \,\middle|\, -\mathbf{R}^T\mathbf{t}\right]$$

to each 2D point $\mathbf{p}$ in the image:

$$\mathbf{x} = \mathbf{T}_{\text{inv}}(d\mathbf{K}^{-1}\hat{\mathbf{p}})$$

[5]We use the THREE library for our web render. THREE and the glTF loader automatically converts between linear RGB and sRGB, but not for meshes that use vertex colors. Thus we need to do this conversion manually for the background.

where: p is the 2D pixel coordinates (u, v) of a point in the current frame; $\hat{\mathbf{p}}$ is $\mathbf{p}$ as homogeneous 2D coordinates $(u, v, 1)$; $\mathbf{x}$ is p projected into 3D world coordinates; $\mathbf{T}_{\text{inv}}$ is the inverse camera pose for which $\mathbf{R}$ is the rotation matrix and $\mathbf{t}$ the translation column vector; d is the depth at the point $\mathbf{p}$; and $\mathbf{K}$ is the camera intrinsic parameter matrix.

The mesh faces are created by running Delaunay triangulation on the 2D points within the dynamic regions of a given frame. The mesh faces are filtered to eliminate stretched or mislabeled faces. Specifically, faces are removed if any vertex's 2D point is more than 2 pixels away from the other vertices or if the depth difference exceeds 10 cm. We then apply two further filtering techniques on the 3D data. First, we apply mesh decimation to reduce the complexity of the mesh, then we use connected components analysis to identify the largest cluster of vertices and discard all other clusters (floaters).

The next step is to map the texture and UV coordinates. We obtain the texture by cropping the RGB frame to the extent of the 2D points (minimum and maximum coordinates along each axis). The UV coordinates are then adjusted to be relative to the texture's top-left corner. Once all objects in the frame have been processed, the mesh data is merged into a single mesh object per frame.

This process is repeated for each frame, taking the respective camera pose into account and storing a frame index. This results in a continuous and coherent mesh representation of the dynamic elements in the video, which are directly used in the final 3D video output.

3.4 Export and Rendering

Our main goal is to render an immersive 3D video in a VR headset. To achieve this, we need to ensure the data format can store all relevant information and can be displayed on a VR headset in real-time.

3.4.1 Data Format

To the best of our knowledge, there is no standard file format for 3D mesh videos. Therefore, we opt to export the 3D video into a folder with three files: one for the foreground meshes, one for the background mesh, and a metadata file. Leveraging the glTF file format, we consolidate the foreground meshes of all frames into a single file, labeling each mesh with its frame index to ensure accurate timing since not all frames may have dynamic elements. We use Draco compression[6] to reduce file size (Table 2) and load times. Applying Draco reduces the average total file size by about 40% to 80 KB per frame, making the 3D video experience more accessible to users.

3.4.2 VR Renderer

To render the videos, we developed a web-based application using WebXR for cross-platform 3D video rendering. Our solution supports basic playback controls and runs at 60 FPS on various devices, including desktops, laptops, mobiles, and VR headsets. For effortless deployment, we offer a Docker image that can be used on local or remote servers, along with a guide for hosting the web app using GitHub Pages[7].

[6] https://google.github.io/draco/
[7] https://github.com/AnthonyDickson/HIVE

Table 2: Compression statistics (mean and standard deviation) for 800 frames across all scenes and configurations listed in Section 4.1. Note that not all frames have foreground elements which is why the mean mesh (frame) count is less than 800

Layer	Meshes	Time (s)	Before (MB)	After (MB)	Compression Ratio
Foreground	572.8	2.1	75.3	59.2	1.29:1
Background	1.0	5.9	34.0	3.3	10.14:1

In the video player, the dynamic meshes are stored in a dictionary keyed by the frame index. The frame index is then used to enable the correct dynamic mesh for each rendering frame. Since the background mesh is a single 3D mesh, it remains constant throughout.

4 Technical Evaluation

4.1 Experiment Setup

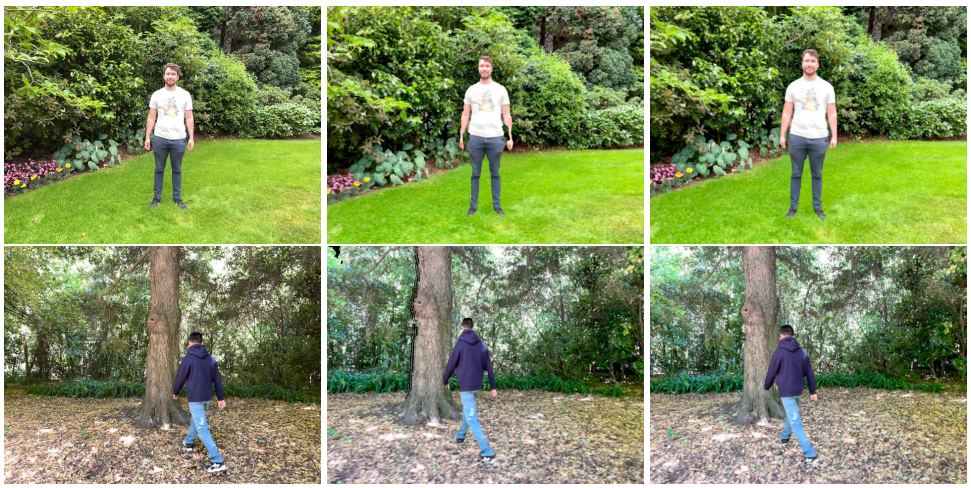
4.1.1 RGB-D Datasets

In our experiments on RGB-D datasets, we use four RGB-D sequences with ground truth depth and camera parameters. Two sequences, 'walking xyz' and 'sitting xyz', are from the TUM RGB-D dataset [40], all of which contain people moving around indoors with a camera being moved along the x, y, and z axes. The 'garden' and 'small tree' sequences were captured outdoors on an iPhone 12 Pro Max with the StrayScanner app and the built-in LiDAR scanner, each containing a single person with the camera either being held stationary or orbiting around them.

We run Mono2VR with three configurations: 'GT', 'CM', and 'EST'. The GT (Ground Truth) configuration utilizes ground truth data for both camera parameters and depth maps. The CM (COLMAP) configuration uses camera parameters estimated with COLMAP and ground truth depth maps. Lastly, the EST (Estimated) configuration makes use of camera parameters estimated with COLMAP and estimated depth maps (i.e., only uses the video frames as input). All sequences are truncated to 800 frames (about 13 seconds at 60 frames per second). We use a key frame threshold of 0.3 and a frame step for COLMAP of 15. All frame data is processed at a resolution of 640×480.

4.1.2 HyperNeRF Dataset

In our experiments on the HyperNeRF dataset [32], we process frames at 540×960 resolution. Due to how Mono2VR scales COLMAP pose data, the provided pose data cannot be used directly for validation. We run COLMAP on all video frames and scale the translation vectors with estimated depth as per Section 3.1.3. We then run Mono2VR on the training frame data and the scaled COLMAP poses from the training frames.



Input GT EST

Fig. 3: Sample of outputs from the Mono2VR system. The rows correspond to outputs from our iPhone datasets 'garden' and 'small tree'. The columns correspond to the reference image, the reconstruction from ground truth sensor data (config = GT), and the reconstruction from monocular video and estimated data (config = EST)

4.1.3 DyNeRF Dataset

In our experiments on the DyNeRF dataset [29], we process frames at 640x480. Since Mono2VR expects the translation vectors to be in the same units as the depth maps (i.e., same scale), we cannot use the provided pose data as-is since it is subject to an unknown scale. We extract the first frame of each video feed, estimate the camera parameters with COLMAP, estimate depth maps for the first frames, and apply our pose scaling approach from Section 3.1.3. We use these scaled camera parameters for positioning the camera at the other camera feeds (01-20) to evaluate Mono2VR on novel viewpoint synthesis.

4.2 Visual Quality

We computed sample outputs from each of the datasets to assess the visual quality (Figure 3). Overall, the outputs from the EST configuration look the most similar to the input frames when viewing from the reference (capture) viewpoint. In all examples, the colors across all configurations differ subtly from the reference frame, most notably the grass in the garden sequence and the leaves in the small tree sequence. This may be caused by color space conversions from RGB to sRGB and vice versa by the renderer. The output from the EST configuration tends to be more complete than the outputs using sensor data for the depth maps (GT and CM). This can be explained by how both the TUM and StrayScanner datasets use LiDAR scanners which have a limited range. This leads to distant parts of the scenes lacking depth data, and this subsequently leads to holes in the background mesh. This is especially noticeable for the examples from the TUM dataset. The depth data from the StrayScanner datasets leads to skinnier foreground meshes. This could be possibly due to the low capture

Fig. 4: Comparison of Mono2VR run on the 'walking xyz' sequence using ground truth depth data with inpainting (left) and without inpainting (right). Inpainting on TUM sequences creates floaters around object boundaries—note the areas around the two people in the scene and the gray colored floaters.

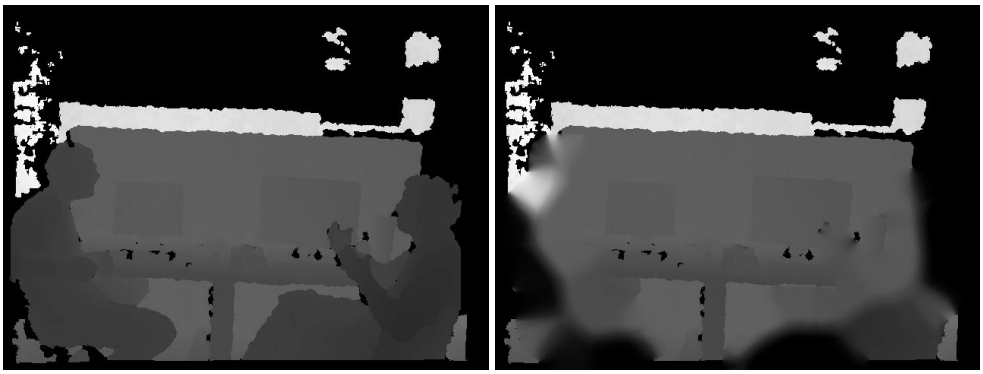
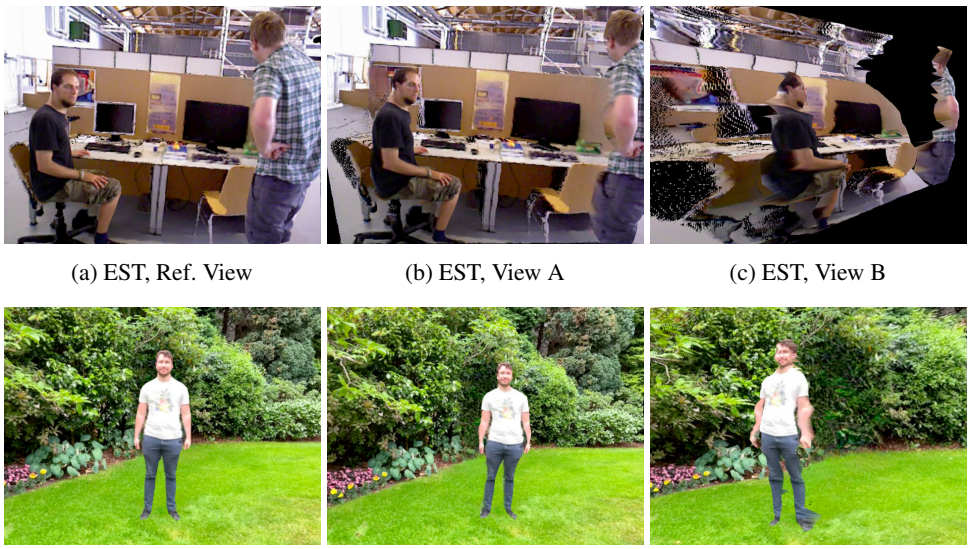


Fig. 5: An example of depth maps from the sitting xyz sequence showing (Left) the ground truth depth map and (Right) the inpainted depth map. Depth values are represented in grayscale with zero as black and max depth (10 m) as white. The inpainted regions where the people were shows how the inpainting algorithm smooths the depth between boundary areas with non-zero values and the boundary areas with zero depth. This is what causes the floaters in the TUM sequences when using inpainting.

resolution of the iPhone's LiDAR scanner (256x192). Additionally, the GT and EST outputs for the TUM sequences have floaters present that seem to form at the edges of a single frame's data and form a path towards the capture camera. These visual artifacts are introduced by inpainting the depth data, and disabling inpainting completely removes the floaters (Figure 4). These floaters are caused by the inpainting algorithm trying to inpaint the region that is bordered by zero and non-zero values. This leads to the inpainting algorithm interpolating a smooth gradient between the non-zero and zero values, which when projected in 3D gives us the floaters that form a path towards zero depth. This is evident when comparing the source and inpainted depth maps, such as in Figure 5.



(a) EST, Ref. View (b) EST, View A (c) EST, View B

(d) EST, Ref. View (e) EST, View A (f) EST, View B

Fig. 6: Sample of outputs from the Mono2VR system comparing reconstructions viewed from the reference viewpoint (Ref. View) and novel viewpoints. The rows show outputs from the 'walking xyz' sequence and the 'garden' sequence. View A shows a small translational change in viewpoint, and View B shows a large rotational view change

Figure 6 shows outputs from the GT and EST configurations and shows the difference between the reference view and two novel viewpoints. The first novel viewpoint (View A) is a translation approximately 10-30 centimeters to the left and the second viewpoint (View B) is a rotation approximately 30°about the y-axis from the reference viewpoint orbiting the scene.

Both configurations handle View A reasonably well and introduce only a small amount of visual artifacts. The TUM examples show the most artifacts after the translation as holes in the background mesh become visible (most notably the office chair in the bottom left corner). The StrayScanner example for the EST configuration shows distortion in the geometry of the person akin to barrel distortion due to inaccuracies in the estimated depth data.

View B proves to be challenging for all configurations and datasets and introduces noticeably more visual artifacts than View A. Only the Kinect sensor of the TUM dataset seems to produce accurate geometry for foreground elements such as people. The examples with the garden sequence handle View B the best. There are no visible holes or tears in the background mesh, unlike the TUM examples. In this case, we see that the background inpainting plausibly fills the hole in the background mesh behind the person. Comparing the examples for the TUM and garden sequences, the depth estimation model seems to better handle 'flat' scenes, scenes where the background could be roughly modeled with planar surfaces at similar depths. Compare the TUM sequence, which has multiple 'layers' (the desk, the divider behind the desk, and the wall in the distance), to the garden scene which has close to one layer—the plants. The visual artifacts seem to mostly be due to inaccurate depth estimation, so improvements in monocular depth estimation will directly improve the outputs of Mono2VR.

4.3 Scaling COLMAP Poses

As shown previously in Figure 2, it is important to scale the COLMAP pose data to match the scale of the depth maps, otherwise the frames will not be correctly aligned. To show that our scaling method can indeed recover metric-scale pose data from COLMAP when using metric-scale depth maps and align frame data accurately, we evaluate pose error against the ground truth pose data provided in our experiment datasets. We use two configurations in the evaluations: CM and EST. The purpose of using the CM configuration is to show how accurate the scaled COLMAP poses are when using ground truth depth data—i.e., to evaluate the scaling method under ideal conditions. The purpose of using the EST configuration is to show the joint effect of our scaling method and estimated depth maps on the accuracy of the scaled COLMAP poses. If the error for the CM configuration is low, that would suggest that our approach to scaling COLMAP poses recovers accurate, metric-scale pose data. If the error for the EST configuration is similar to that of the CM configuration, that would suggest that the estimated depth maps from DPT can be used in place of ground truth depth maps to recover accurate, metric-scale pose data.

We compare the scaled COLMAP pose data against the ground truth pose data with the Relative Pose Error (RPE) and Absolute Trajectory Error (ATE) metrics. We calculate the RPE for translation and rotation between adjacent frames:

$$\text{RPE}_r = \sqrt{\sum_{i=1}^{N-1} \angle(\mathbf{R}_{i+1}^T \mathbf{R}_i)^T (\hat{\mathbf{R}}_{i+1}^T \hat{\mathbf{R}}_i)^2} \tag{4}$$

$$\text{RPE}_t = \sqrt{\sum_{i=1}^{N-1} ||\text{tran}((\mathbf{P}_{i+1}^{-1} \mathbf{P}_i)^{-1} (\hat{\mathbf{P}}_{i+1}^{-1} \hat{\mathbf{P}}_i))||_2^2} \tag{5}$$

where: $\angle$ converts a rotation matrix to axis-angle representation and gets the rotation angle; and $\text{tran}(...)$ extracts the translation vector from a pose. We calculate ATE in a similar way to the authors of [50], scaling the predicted trajectory to the ground truth trajectory:

$$\text{scale} = \frac{\sum_{i=1}^{N} \mathbf{T}_i \odot \hat{\mathbf{T}}_i}{\sum_{i=1}^{N} \hat{\mathbf{T}}_i \odot \hat{\mathbf{T}}_i} \tag{6}$$

where: T_i is the ith translation vector in the ground truth trajectory; $\hat{T}_i$ is the ith translation vector in the estimated trajectory; and $\odot$ denotes the element-wise product. The ATE metric is then calculated as:

$$\text{ATE} = \sqrt{\sum_{i=1}^{N} (\mathbf{T}_i - \text{scale} \times \hat{\mathbf{T}}_i)^2} \tag{7}$$

Our results (Table 3) indicate that the depth data estimated with DPT [41] can be used to recover metric-scale pose data from COLMAP with comparable accuracy to that of the ground truth depth data. The pose data scaled with the estimated depth data compared to the pose data scaled with ground truth depth on average had: about the same rotational RPE (less than 0.01°difference), 0.06 cm (21%) higher RPE and 0.08 cm (2%) lower ATE. The low error further suggests that the scaled COLMAP poses can accurately align the frame data.

Table 3: Comparison of the ground truth trajectory and the trajectories from COLMAP scaled with ground truth depth maps (Config = CM) and scaled with estimated depth maps (Config = EST). For each metric, lower is better.

Dataset	Config	RPE_r (°)	RPE_t (cm)	ATE (cm)
walking xyz	CM	0.57	0.52	3.30
	EST	0.57	0.60	2.50
sitting xyz	CM	0.44	0.32	3.07
	EST	0.44	0.41	3.66
garden	CM	0.06	0.11	1.57
	EST	0.06	0.15	1.56
small tree	CM	0.10	0.16	4.84
	EST	0.10	0.19	4.85
Mean	CM	0.29	0.28	3.22
	EST	0.29	0.34	3.14
	All	0.29	0.31	3.17

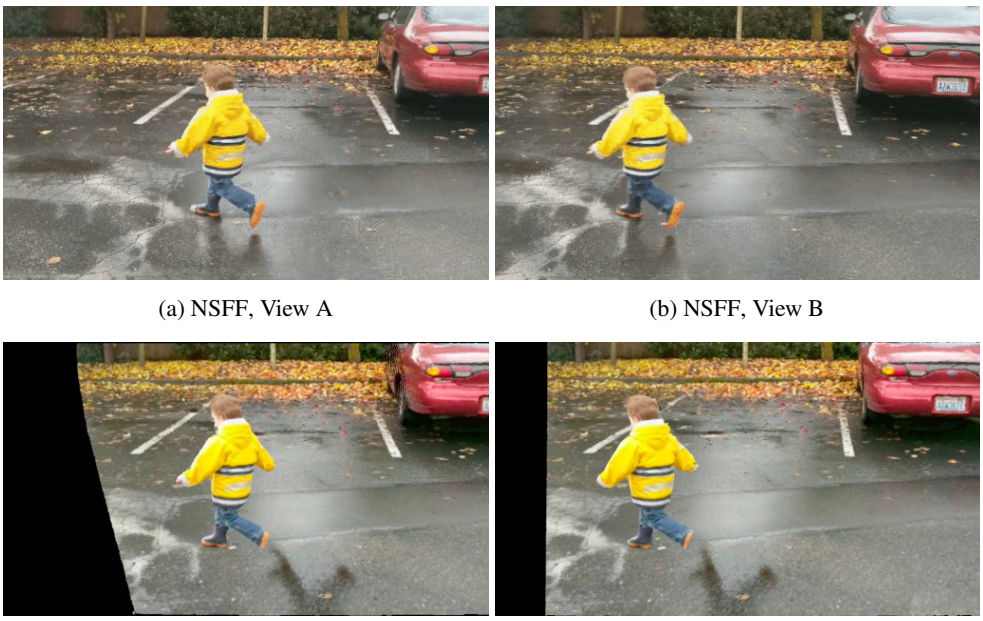
4.4 Mono2VR and NSFF

We compare Mono2VR against Neural Scene Flow Fields (NSFF) [8], a NeRF technique that has been adapted for video. Both methods are similar in that they both work on monocular RGB video and are restricted to camera movements that COLMAP can work with. NSFF generally produces more complete and accurate renders than Mono2VR, but has disadvantages when it comes to hardware requirements, compute time and interactivity.

Both methods can produce realistic renders from novel viewpoints close to the reference viewpoint (Figure 7). The most notable visual artifacts in the Mono2VR output are the black borders due a lack of mesh data and static reflections on the ground due to the use of a static background mesh.

NSFF has hardware requirements that goes beyond the typical computer, beyond even enthusiast computers that have dedicated graphics hardware. We were unable to independently verify the VRAM usage of NSFF since we were unable to run it on our own hardware. However, based on the information from the paper and the official code repository, we estimate VRAM usage to be between 32-64 GB for a 75 clip processed at a resolution of 512x288. In comparison, Mono2VR uses about 4 GB when processing the same clip at a resolution of 640x360. Just the memory usage alone greatly restricts access to NSFF. When looking at the Steam Hardware and Software Survey: December 2023[8], we find that 53% of enthusiast computers have a graphics card capable of running Mono2VR (GTX 1060 6 GB equivalent or better) and no more than 10% of enthusiast computers could *potentially* run NSFF (RTX 2080 Ti 11 GB equivalent or better). We say 'potentially' because generally consumer PCs only have one GPU, whereas NSFF requires at least two GPUs with 16 GB or more VRAM[9], or 4 GPUs with 11 GB or more VRAM. These issues with high VRAM usage and subsequent hardware

[8] https://store.steampowered.com/hwsurvey/videocard/, accessed 30 Jan, 2024
[9] only 3.3% of PCs in the survey had at least one GPU with 16 GB or more of VRAM.



(a) NSFF, View A

(b) NSFF, View B

(c) Mono2VR, View A

(d) Mono2VR, View B

Fig. 7: A comparison of novel viewpoints rendered with NSFF and Mono2VR from frame 10 of the kid running sequence. The black pixels in the Mono2VR samples are due to no mesh data. For NSFF, we chose frames from the example GIF (https://github.com/zhengqili/Neural-Scene-Flow-Fields/blob/main/demo/sti.gif) and chose the pair of frames that are approximately the most distant. A live demo of Mono2VR's 3D video for the kid running sequence is available at https://anthonydickson.github.io/HIVE_Renderer

requirements of NSFF are only exacerbated when the video length is increased beyond the one second (75 frames) in this example clip. The amount of video that the underlying MLP model can accurately represent is restricted by its size, but increasing the size of the MLP model increases the memory usage rapidly due its non-linear scaling. This makes it increasingly difficult to apply NSFF to longer video clips. In contrast, the hardware requirements for Mono2VR are fixed regardless of the length of the input video.

The computational complexity of NSFF results in long compute times even for short clips. For the 'kid running' clip of 75 frames, the author reported that it took about 2 days to train. In contrast, Mono2VR does not require any test-time training and processes the same clip in about two and half minutes. The compute time of NSFF is affected by video length in a similar way to memory usage, as the size of the MLP increases the compute time will likely increase non-linearly. Mono2VR scales linearly, except for camera parameter estimation with COLMAP, which appears to scale non-linearly.

The high per-frame processing times of NSFF present a barrier to usage in interactive applications, especially VR. It takes 6 seconds for NSFF to render a single frame at a resolution of 512x288. 60 frames per second (0.016 seconds per frame) is the ideal minimum frame rate for interactive applications for smooth movement/animation and keeping camera movement

in sync with head movements in VR. It is especially important to maintain this minimum frame rate in VR to avoid causing or worsening motion sickness [51]. When tested on the 'kid running' sequence, Mono2VR processed the video at a resolution of 640x360 and rendered the mesh video at up to 120 frames per second (0.008 seconds per frame) across a range of devices (desktop PC w/ RTX 3080, M1 MacBook Pro, and iPhone 13 Pro).

Overall, NSFF produces the highest quality synthesized novel-viewpoint, however it requires expensive hardware, takes a long time to run, and slow render times make it unsuitable for interactive applications. Mono2VR can run on consumer hardware in a relatively short amount of time, and owing to the choice of triangle meshes as the 3D model representation renders at interactive frame rates.

4.5 Evaluation on HyperNeRF

Table 4: Quantitative comparisons on the HyperNeRF dataset [32]. Reported figures for Mono2VR and 4D-GS are from experiments run by us; figures for other methods are from respective publications. Hardware access for Mono2VR and 4D-GS is calculated from observed VRAM usage; for others, from reported hardware. Processing times for Mono2VR and 4D-GS include time for COLMAP. †Excludes regions with no data in the rendered frame from Mono2VR. *Performance and hardware requirements are as reported in [8], image metrics in [32]. **HyperNeRF VRAM is VRAM usage per TPU multiplied by number of TPUs.

Sequence	Time	Frames Per Minute	Render FPS	Storage (MB)	VRAM (GB)	GPU	Hardware Access	MS-SSIM	PSNR	LPIPS
Mono2VR (Ours)	14m	19	**60+**	35	**4.1**	**1x RTX 3080 (10 GB)**	**53%**	.315	9.5	.668
Mono2VR (Ours)†								.578	16.5	.367
NSFF [8]*	48h00m	0.04	0.16	-	32	2x Tesla V100 (16 GB)	$\leq 10\%$	**.917**	23.2	.174
HyperNeRF [32]**	8h00m	0.14	-	-	128	4x TPU v4 (32 GB)	$< 1\%$	.811	22.2	**.153**
NeRFPlayer [33]	5h30m	0.9	-	-	48	1x RTX A6000 (48 GB)	$< 1\%$	-	**30.3**	-
TiNeuVox-S [34]	**10m**	**27**	-	-	24	1x RTX 3090 (24 GB)	1.5%	.813	23.4	-
4D-GS [9]	1h07m	4.0	14	**34**	4.3	**1x RTX 3080 (10 GB)**	**53%**	.865	25.8	.323

We evaluate Mono2VR on the HyperNeRF dataset [32] and compare our approach to NSFF [8], HyperNeRF [32], NeRFPlayer [33], TiNeuVox-S [34] and 4D-GS [9] with regards to performance, hardware requirements, and visual quality (Table 4). Sample outputs from Mono2VR as shown in Figure 8.

One issue with Mono2VR and the key frame sampling approach is that it typically only uses one frame for reconstructing the background mesh, leading to regions with no data (e.g., Figure 8, bottom row). We additionally report the image similarity metrics after excluding regions in the rendered frame that have no mesh data to evaluate the quality of the present mesh data.

The HyperNeRF dataset proves challenging for Mono2VR and highlights the limitations of our approach to instance segmentation. The HyperNeRF dataset contains a scene where the only moving object is not a person (the 3D Printer sequence) and the sequences have people only partially in frame handling an inanimate object. Mono2VR is designed for scenes focused on people and only people are considered for inclusion in the dynamic reconstruction. The dynamic elements in these sequences are therefore ignored by Mono2VR.

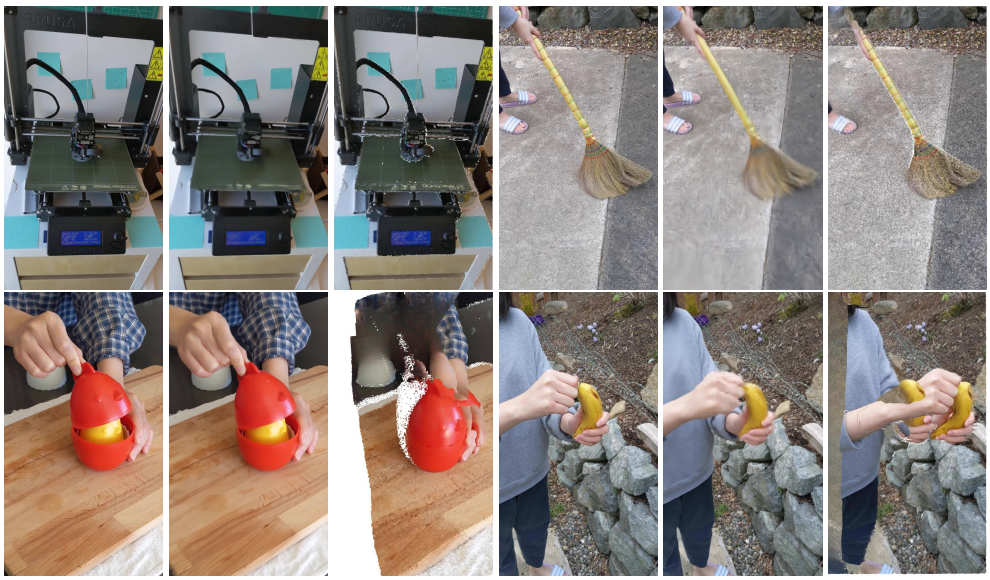


Fig. 8: A comparison of 4D-GS and Mono2VR on the HyperNeRF dataset. The top row show the results from frame one and the bottom row show the results for frame five

It is important to mention that the more recently proposed Gaussian splatting method 4D-GS [9] has lower computational requirements compared to the NeRF-based methods for view synthesis, but our Mono2VR retains an advantage in processing time and render frame rates and is about four times faster than 4D-GS. The biggest challenge in using 4D-GS for interactive applications, and especially immersive VR applications, is its low render framerate of about 14 FPS on the HyperNeRF dataset using our experimental setup (RTX 3080). On the DyNeRF dataset [29], 4D-GS rendered at around 18 FPS. These frame rates are for monocular rendering, and would likely be substantially lower for stereo rendering in VR. In comparison, Mono2VR (M2VR) renders at a steady 60 FPS on typical displays and at 120 FPS on high refresh rate displays even on mobile devices, largely owing to our use of standard 3D mesh formats. Perhaps converting the 4D-GS outputs into our mesh format, similar to [52], could combine the high visual quality of 4D-GS and the fast rendering of Mono2VR. Recent work such as MoDGS [53] looks promising for achieving 60+ FPS rendering with Gaussian Splatting.

4.6 Ablation Study

We evaluate Mono2VR on the DyNeRF dataset [29] to see how components of Mono2VR affect the outputs.

Multicam uses the scaled multi-camera pose data from the above process along with the video from the test video (cam00). This gives us a baseline to compare the other configurations against. **Monocular** only uses the test video and uses the Kinect sensor intrinsic matrix[10] as

[10]Using a simple pinhole camera model, the Kinect intrinsic parameters we use are: fx=fy=580, cx=319.5, cy=239.5.

an estimate. This gives us an idea on how Mono2VR performs with solely monocular data. **BundleFusion** shows how BundleFusion [54], a TSDF RGB-D fusion method that builds upon KinectFusion [27], compares to TSDF Fusion with scaled COLMAP poses for reconstructing the background mesh from estimated depth data. **Compression** follows the same configuration as Monocular, however the mesh data is compressed with Draco and uncompressed before rendering. This shows how much compression affects the quality of the outputs. **No CC Analysis** follows the same configuration as Monocular, however the Connected Components (CC) analysis filtering is disabled (this affects the foreground mesh data). **No Inpainting** follows the same configuration as Monocular, however the mesh data is created without inpainting the holes in the background.

Due to Mono2VR only using a single video feed to reconstruct the scene, there are unobserved regions of scene Mono2VR cannot possibly reconstruct that are visible from the other camera feeds (01-20). We also include the image similarity metrics calculated only on the regions that Mono2VR observed to give an idea of the quality of the reconstructed mesh data without considering the regions with no corresponding input data. To create the masked image, we mask the input video frames by painting white the regions that correspond to the regions in the rendered frame that are missing mesh data ($RGB = [255, 255, 255]$). Note the difference between results for the non-masked and masked images for the cameras 01-20. This shows the impact of unobserved regions on the scores of novel viewpoints.

The results in Table 5 show the relative gain/loss in render quality between each configuration. Overall, the biggest changes are observed when changing the background reconstruction algorithm to BundleFusion and when removing background inpainting (Figure 9). We initially considered using BundleFusion for background reconstruction but found it was not suitable for use with estimated depth data. We observe that BundleFusion results in blurrier textures and gaps in the background mesh. Removing background inpainting has a large impact on visual quality. Although, this is likely due to our aggressive mask dilation creating a larger than necessary borders around foreground elements. We note that removing the connected components analysis filtering and adding compression have a neglible impact on visual quality. The connected components analysis filtering is most effective when background is confused for foreground, which does not seem to happen much with this dataset.

5 User Evaluation

We conducted a user study to evaluate the user experience and visual quality of Mono2VR. As there are currently limited options available for rendering videos in VR, we compared our method to a 2D video rendered in VR as well as 3D videos created with ground truth depth. This study has received ethical approval from the University of Otago's ethics committee and followed health and safety precautions such as providing hand sanitizers and using disposable VR Covers for the headsets.

5.1 Study Design and Apparatus

We designed a within-subject user study to explore the use of 2D and 3D videos in more detail. The videos used in the study capture a variety of scenes, including outdoor and indoor scenes with varying depths. All videos have one human subject performing different actions. The starting location of each video is manually calibrated to the "sweet spot" for minimal distortion

Table 5: Ablation study on render quality. Mono2VR is run on camera feed 00 and camera feeds 01-20 are used for novel viewpoint synthesis. See Section 4.6 for details on the configurations. †Excludes empty regions

Config	Camera Feed	SSIM	PSNR	LPIPS
Multicam	00	.848	26.0	.163
	01-20	.453	9.4	.476
	01-20†	.628	18.8	.208
Monocular (Ours)	00	.860	25.9	.188
	01-20	.370	8.3	.619
	01-20†	.588	16.4	.304
BundleFusion	00	.658	16.1	.335
	01-20	.391	8.1	.627
	01-20†	.623	16.9	.308
Compression	00	.860	25.9	.188
	01-20	.370	8.3	.619
	01-20†	.588	16.4	.304
No CC Analysis	00	.862	26.0	.191
	01-20	.370	8.3	.619
	01-20†	.588	16.4	.303
No Inpainting	00	.843	18.5	.220
	01-20	.369	8.0	.629
	01-20†	.614	16.5	.283

Input (cam06) Monocular No Inpainting BundleFusion

Fig. 9: Example outputs from the ablation study of Mono2VR run on the DyNeRF dataset [29]. We show the input frames, the outputs from our method, Mono2VR (Monocular), and the configurations that show the largest change in visual quality, 'No Inpainting' and 'BundleFusion'

Fig. 10: Scene used for the first part of the user study. (**Left**) 2D-Video, (**Center**) 3D-GT, (**Right**) 3D-EST.

when viewing from novel viewpoints. The study had two parts, the first part compared the user experience and presence between 2D video (2D-Video), 3D ground truth depth video (3D-GT), and 3D estimated depth (3D-EST) video. The second part compares the visual quality between just 3D-GT and the 3D-EST. We used 2D videos as a baseline since these are the standard way of watching videos in VR (e.g. YouTube VR). We decided against using RGB-D and MPI videos because they induced motion sickness in preliminary tests and would not create a pleasant experience for users. Instead, we decided to use 3D ground truth videos as a baseline since they they use sensor input for depth and would provide more insights.

The task for the participants was to explore the videos. The first part of the study used the same video (Figure 10) for all three conditions (2D-Video, 3D-GT, 3D-EST). The order of the conditions was randomized using Latin Square to minimize the influence of learning and order effects. The videos were viewed in the Firefox web browser through the VR headset and Oculus Link Desktop. The second part of the study uses 3 sets of videos (Figure 11), and video and condition order were also randomized. The two conditions (3D-GT and 3D-EST) for the same video are shown successively for better comparison.

We hosted the videos locally on a desktop with an i9 processor, 64GB of RAM and an NVIDIA Quadro RTX5000 graphics card. A Meta Quest 2 with a disposable VR Cover was connected via the Link Cable to the desktop workstation. Participants were asked to sit on a swivel chair in which they could rotate themselves and look around. All questionnaires were done on paper by the participant, except for the questions in part 2 where the study instructor wrote down the participant's verbal answers.

5.2 Participants

We recruited 24 participants to balance the Latin Square randomization for the user study conditions. Participants were recruited from the University of Otago through advertisements and word of mouth, and were only required to be between 18-65 years old. In total, 24 participants (15 male, 8 female, and 1 Other) aged between 18 and 34 ($\bar{x} = 22.04$, $\sigma = 4.50$) participated in our study. Among them, 18 participants knew what of VR and had prior experience, while 5 participants had heard of it but had no experience. One participant was new to the term VR.

5.3 Procedures

Upon arrival, participants were provided with an information sheet detailing the objectives, procedures, and expectations of the study. They were given time to review the information

Fig. 11: Scenes used for the second part of the user study. (**Left**) Input video, (**Middle**) 3D Video created with Ground Truth data (RGB-D + poses captured with StrayScanner app), (**Right**) 3D Video created from video input only.

sheet and invited to fill in a consent form to confirm their voluntary participation. Additionally, they were asked to complete a demographic questionnaire, which collected information on their age, gender, vision status, and any relevant experience with VR.

5.3.1 Briefing

After completing the initial paperwork, participants were briefed on the structure and flow of the user study. We explained the study goals and gave an overview of the two parts of the study. Throughout the briefing, participants were encouraged to ask questions and clarify any aspects of the study they found unclear. The user study began when the participant was ready and had no further questions.

5.3.2 Part 1: User Experience and Presence

Participants first put on the VR headset and adjusted for a good fit to ensure they could see clearly. If they still had trouble seeing clearly, the interpupillary distance (IPD) was changed. Participants were then presented with the first video. They were given ample time to watch the video, which looped upon completion, allowing them to experience the condition for as long as they felt necessary.

When participants were finished viewing the video, they removed the VR headset and completed the User Experience Questionnaire (UEQ) [55] and the Igroup Presence Questionnaire (IPQ) [56]. These questionnaires aimed to capture their impressions of the user experience

and their sense of presence in the virtual environment for each condition. After completing the questionnaires for the first condition, participants moved on to the next condition, following the same procedure of putting on the VR headset, watching the video, and filling out the questionnaires. This process was repeated for all three conditions in part one of the study.

5.3.3 Part 2: Depth Perception and Visual Quality

Participants kept their VR headset on for the entire duration of this part of the study. The focus of this part was to evaluate the depth perception and visual quality of the videos, and the participants were guided through a series of prompts for each video. They were asked to rate the depth perception, visual quality, and presence of visual artifacts in the video using a 7-point scale, with 1 representing very low and 7 representing very high. To begin, the first condition of the randomized video was played. After the participant viewed and assessed the video, they answered the three prompts in sequence. This is repeated for the second condition of the same video. The same procedure was followed for the remaining two videos in part two of the study.

5.3.4 Debriefing

Once the second part of the study was completed, participants filled in a questionnaire. This questionnaire asked about their preferences and opinions about using 2D and 3D videos for reliving past memories, and under what circumstances they might choose one format over the other. Additionally, participants were asked if they would recommend such an experience to their friends and family. Participants were encouraged to provide any further feedback they had regarding the overall study.

5.4 Results

We analyzed the results using R and Excel following the instructions for each of the used questionnaires. We performed the Shapiro-Wilk normality test and where appropriate used repeated measure ANOVA and t-tests or Friedman and Wilcoxon.

5.4.1 UEQ

For most UEQ ratings, the scores for all three conditions were in a neutral range (scores for efficiency, novelty and stimulation being in the range of -0.8 – 0.8). The 2D video format received lower scores in the Stimulation (-0.323) and Novelty categories (-0.715) compared to the 3D-GT condition (0.48 (S) and 0.72 (N) and the 3D-EST condition (0.634 (S) and 0.75 (N)). This was anticipated, given that 2D videos are a common and less engaging format compared to 3D videos.

The 2D video format showed higher scores than the other video types in the areas of perspicuity (1.719 (2D-Video), 1.15 (3D-GT), 1.524 (3D-EST)) and dependability (1.01 (2D-Video), 0.55 (3D-GT), 0.85 (3D-EST)). For perspicuity, all conditions are in the range of a positive evaluation (> 0.8 [55]), and for dependability both 2D video and 3D-EST are within the positive range. Overall, the scores could be attributed to the fact that 2D videos are currently the most widely used and familiar format, making them easier to understand and more reliable for users. Overall, we only found significant differences between the 2D video

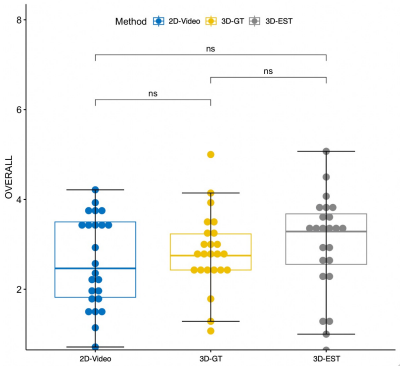


Fig. 12: Comparison of the overall IPQ scores of the 2D-Video, the 3D Ground Truth Depth (3D-GT) and 3D Estimated Depth (3D-EST) conditions.

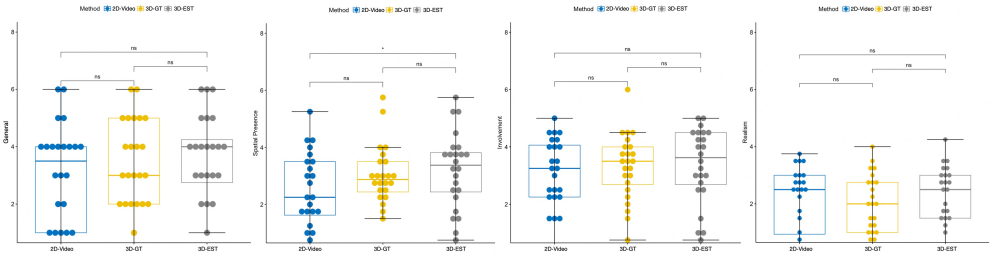


Fig. 13: Comparison of the IPQ scores of the 2D-Video, 3D Ground Truth Depth (3D-GT) and 3D Estimated Depth (3D-EST).

condition and the 3D video conditions for novelty (2D-Video-3D-GT: p=0.029, 2D-Video-3D-EST p=0.0113) and stimulation (2D-Video-3D-GT: p=0.0001, 2D-Video-3D-EST p=0.0002) with measurable lower scores for the 2D video.

5.4.2 Presence

We employed the Immersive Presence Questionnaire (IPQ) to analyze presence [56]. The dependent variable was the IPQ score. The independent variable was represented by the VRVideo method 2D-Video, 3D-GT, and 3D-EST. We computed the overall IPQ presence score [56–58] (Figure 12), along with the IPQ subscale scores. In accordance with the IPQ data analysis guidelines[11], we transformed the IPQ scales to span a range of 0 to 6.

Descriptive statistics reveal that the 2D-Video method (mean = 2.61, std =1.02, median = 2.46) and 3D-GT method (mean = 2.83, std =0.85, median = 2.75) exhibit similar medians for the overall presence score, while the median for 3D-EST is higher (mean = 3.01, std = 1.1, median = 3.29). We applied a Shapiro Wilk test and a p-value = 0.343 indicates that we can assume normality. Using repeated measures ANOVA, we identified a statistically significant difference in the IPQ scores depending on the method, $F = 3.602$, p = 0.0352. The result is

[11] http://www.igroup.org/pq/ipq/data.php

significant at $p < .05$. We then performed a t-test (with Holm correction) for post-hoc analysis, which indicated no significant differences between the conditions (Table 6).

Table 6: Overall Presence: p values t-test.

	2D-Video	3D-GT
3D-GT	0.33	-
3D-EST	0.15	0.33

Cohen's d indicates small effect sizes (2D-Video-3D-GT d = 0.206 (small), 2D-Video-3D-EST d = 0.402 (small), and 3D-GT-3D-EST d = 0.281 (small)).

We also analyzed the IPQ subscales (General presence (G), Spatial presence (SP), Involvement (INV), and Realism (REAL)) for a more detailed analysis (Figure 13). For Spatial Presence, the Shapiro-Wilk (SW) normality test indicated normal distribution (p=0.5033), so we used a repeated measure ANOVA. We found a significant effect of the video method for Spatial Presence (SP) (F(2, 46) = 3.452, p = 0.0401) and further analyzed the data with a t-test (Holm). The analysis showed that there is a significant difference between the 2D-Video condition and 3D-EST (Table 7).

Cohen's d displays small to moderate effect sizes (2D-Video-3D-GT d = 0.388 (small), 2D-Video-3D-EST d = 0.627 (moderate), and 3D-GT-3D-EST d = 0.314 (small)). We did not find a significant effect of video type on G (Friedman p = 0.70, SW p=0.0026), nor on SP (p = 0.137), nor on INV (Friedman p-value = 0.461, SW p=0.0405), nor on REAL (Friedman p = 0.102, SW p = 0.011). The results indicate that there is an overall effect on presence and that there is a significant increase in spatial presence (SP) between 2D-Video and our 3D-EST method.

5.4.3 Depth Perception and Visual Quality

Depth perception was generally high across both 3D-GT and 3D-EST conditions (Figure 14, 3D-GT: m=4.82, std=0.98, 3D-EST: m=5, std=1.13). Interestingly, we could not find a significant difference between the 3D-EST condition, which relied on estimated depth, and the 3D-GT condition which utilized LiDAR sensor data from the iPhone 13 Pro (t-test: p=0.25). This suggests that the depth estimation techniques used in the 3D-EST condition created convincing depth perception for the participants.

Additionally, the visual quality of the 3D-EST (m=3.18, std= 1.29) condition was rated better than the 3D-GT condition (m=3.58, std= 1.16, t-test p=0.029) which might be due to the smoother estimated depth data. We did not find any significant differences in the number of artifacts, but both conditions showed more than the average amount of artifacts (3D-GT: m=4.53 std=1.01, 3D-EST: m=4.29, std=1.02, t-test: p=0.13). The visual quality of 3D videos has room for improvement as the general rating was around midpoint (3.5).

Table 7: Spatial presence (SP): p values t-test.

	2D-Video	3D-GT
3D-GT	0.14	-
3D-EST	0.016*	0.14

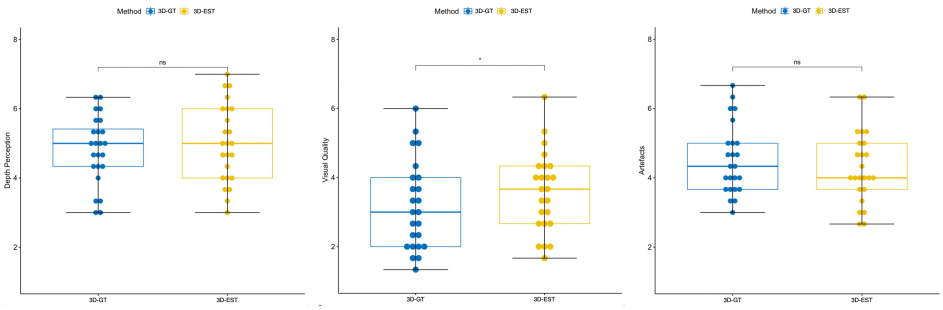


Fig. 14: Depth perception, visual quality and amount of artefacts of 3D-GT and 3D-EST.

When participants were asked if they would prefer the 2D video or 3D video, 18 of the 24 participants preferred the 3D video (75%). Participants also rated an average of 4.96 out of 7 when asked if they would recommend such 3D video. The results show that participants have a positive experience with 3D videos. However, it is important to mention that 50% of the participants mentioned video quality as a factor to be improved.

5.5 Discussion

Overall, the study showcased the potential of 3D videos as a viable medium for reliving past memories, with most participants expressing their acceptance of this presentation format. The advantages of 3D videos, such as enhanced spatial presence and improved visual quality, make the experience more immersive. However, the study also highlighted certain challenges, such as the presence of visual artifacts and visual quality, which could impact the user experience.

5.5.1 Depth Perception

The 3D videos contributed to an above-average rating of the depth perception, which in turn potentially led to a heightened sense of spatial presence for the viewers. Participants, including P4, P9, P11, P13, P21, and P23, reported feeling as if they were ”being there” or ”in the moment” while watching both the 3D videos. It is important to note that the 3D-GT condition did not render pixels beyond the range of the LiDAR sensor. The 3D-EST condition on the other hand renders every pixel, creating a more complete scene, which might have contributed to a slightly improved depth perception.

5.5.2 Visual Quality

The results reveal that 3D videos enhance spatial presence but also highlight a need for improvement in the quality of the 3D videos, as highlighted by 50% of the participants. Participants P14, P16, and P17 commented on the artifacts they encountered along the edges of the subject, such as ”shakiness” or a ”saw-tooth effect.” These visual artifacts can be attributed to the discrepancy between the user’s viewpoint and the original camera position where the video was captured. The distortion they perceive becomes more pronounced as the user moves their head further from the camera’s original position. This is an inherent challenge when generating 3D videos from monocular input. The background mesh of the video could also

be improved. Due to the fusion of images, the visual quality is not as clear as the 2D video, which P14 and P17 mentioned. P10 also mentioned that they wanted a full 360°experience. However, this is not possible with monocular video.

5.6 Limitations

While our user study indicates some promising results, there are a few limitations that need to be considered.

The user study used an old version of Mono2VR, hence why there is an observable difference in the quality of the outputs in this section (Figure 10, Figure 11) and the previous section (Figure 3). The difference between these sample outputs is due to the following three changes. Firstly, the old version did not perform gamma correction, leading to incorrect colors. Secondly, the old version uniformly sampled frames for the background reconstruction, often leading to blurry textures. Thirdly, the old version used sub-optimal settings for the TSDF step, leading to background meshes that lacked detail and blurry textures.

One of the limitations is the relatively low resolution of the LiDAR sensor used for capturing depth information. The limited resolution may result in less accurate depth maps, which can impact the overall visual quality of the 3D-GT videos. This is particularly obvious for the detection of the subject in the video. The edges of the human subject in the 3D-GT videos are often pixelated. This could be improved by smoothing the edges to provide a similar experience to the 3D-EST videos where every pixel is used for depth estimation and results in smoother boundaries. Despite there being higher resolution LiDAR scanners available, our target is to use consumer hardware that is widely available and suitable for casual video capture.

Another limitation in our study is that the 3D videos have a "sweet spot" around the camera's position during video capture. Deviation from this position results in visual distortions and artifacts. We limited the impact of this by placing participants in a swivel chair.

Lastly, the quality of the captured videos and the complexity of the background scenes can also affect the performance of the 3D video generation pipeline. Complex background scenes can pose challenges for the depth estimation and reconstruction algorithms, leading to less accurate 3D video generation. A good example was the moving leaves in our courtyard video, which made the leaves' texture appear blurry. In addition, the video capture must have enough movement so that it works as input for the SfM methods. This requirement could be addressed by using a spherical SfM approach [59, 60].

In the study, we occasionally experienced an Oculus Link bug where videos were not loaded initially (the screen in the headset showed an hourglass symbol) until the user removed the headset and put it back on again. As this did not happen during the trials but just while starting a trial, we are confident that this was not a confounding factor.

6 Conclusion and Future Work

In this work, we presented Mono2VR, a pipeline for creating and exploring immersive 3D video experiences from a monocular video. Mono2VR automatically handles the estimation of the camera parameters, depth maps, 3D reconstruction of dynamic foreground and static background elements on consumer hardware within a reasonable time frame. We showed that with our pipeline, we are able to render VR videos on computers or VR headsets with interactive framerates, providing an immersive and engaging experience for users.

Through our technical evaluation, we demonstrated the key advantages of Mono2VR relative to NeRF and 3D-GS methods. Mono2VR can run on a greater proportion of existing computers with consumer hardware, is significantly faster to run, achieves significantly higher framerates, and provides direct support for VR rendering. Our user study served as an evaluation of the technology, identifying its strengths and areas requiring further improvement. Participants provided valuable feedback on aspects such as depth perception, visual quality, and presence, indicating the potential of our pipeline in delivering an engaging and immersive experience. Nevertheless, the study also revealed limitations, such as visual artifacts and the need for higher-quality output.

The insights from the user study and technical evaluations will help guide our future work, as we continue refining the pipeline in iterations and addressing identified shortcomings. The main areas we consider for future work are as follows. A more general approach to separating foreground and background is needed to improve the reconstruction quality on scenes where dynamic elements other than just people, perhaps using optical flow, and will help improve the outputs for datasets like HyperNeRF [32]. Handling effects such as shadows and reflections would also help improve the visual quality. The accuracy of single image depth estimation models leads to inaccurate geometry and inconsistencies between frames. Therefore, depth estimation models that can leverage video data should be investigated. There is also a need for reducing the compute time for depth estimation models; the model we use [41] takes up close to a third of the runtime of Mono2VR.

Overall, Mono2VR opens new areas of research targeting novel ways of replaying, editing and interacting with VR videos. We hope that our open source software framework will support future research in this direction.

References

[1] Richardt, C., Pritch, Y., Zimmer, H., Sorkine-Hornung, A.: Megastereo: Constructing high-resolution stereo panoramas. In: Conference on Computer Vision and Pattern Recognition (CVPR), pp. 1256–1263 (2013). https://doi.org/10.1109/CVPR.2013.166 . http://richardt.name/megastereo/

[2] Richardt, C., Hedman, P., Overbeck, R.S., Cabral, B., Konrad, R., Sullivan, S.: Capture4VR: From VR photography to VR video. In: SIGGRAPH Courses (2019). https://doi.org/10.1145/3305366.3328028 . https://richardt.name/Capture4VR/

[3] Bertel, T., Yuan, M., Lindroos, R., Richardt, C.: OmniPhotos: Casual 360° VR photography. ACM Transactions on Graphics **39**(6), 266–112 (2020) https://doi.org/10.1145/3414685.3417770

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

[5] Mildenhall, B., Srinivasan, P.P., Ortiz-Cayon, R., Kalantari, N.K., Ramamoorthi, R., Ng, R., Kar, A.: Local light field fusion: Practical view synthesis with prescriptive sampling guidelines. ACM Trans. Graph. **38**(4) (2019) https://doi.org/10.1145/3306346.3322980

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

[7] Kerbl, B., Kopanas, G., Leimkühler, T., Drettakis, G.: 3d gaussian splatting for real-time radiance field rendering. ACM Transactions on Graphics **42**(4), 1–14 (2023)

[8] Li, Z., Niklaus, S., Snavely, N., Wang, O.: Neural scene flow fields for space-time view synthesis of dynamic scenes. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 6498–6508 (2021)

[9] Wu, G., Yi, T., Fang, J., Xie, L., Zhang, X., Wei, W., Liu, W., Tian, Q., Wang, X.: 4d gaussian splatting for real-time dynamic scene rendering. arXiv preprint arXiv:2310.08528 (2023)

[10] Serrano, A., Kim, I., Chen, Z., DiVerdi, S., Gutierrez, D., Hertzmann, A., Masia, B.: Motion parallax for 360° rgbd video. IEEE Transactions on Visualization and Computer Graphics (2019)

[11] Balasubramanian, S., Soundararajan, R.: Prediction of discomfort due to egomotion in immersive videos for virtual reality. In: 2019 IEEE International Symposium on Mixed and Augmented Reality (ISMAR), pp. 169–177 (2019). https://doi.org/10.1109/ISMAR.2019.000-7

[12] Dickson, A., Shanks, J., Ventura, J., Knott, A., Zollmann, S.: Vrvideos: A flexible pipeline for virtual reality video creation. In: 2022 IEEE International Conference on Artificial Intelligence and Virtual Reality (AIVR), pp. 199–202 (2022). https://doi.org/10.1109/AIVR56993.2022.00039

[13] Hartley, R., Zisserman, A.: Multiple View Geometry in Computer Vision. Cambridge University Press, ??? (2003). http://dl.acm.org/citation.cfm?id=861369

[14] Horry, Y., Anjyo, K., Arai, K.: Tour Into the Picture: Using Spidery Mesh Interface to Make Animation from a Single Image", pp. 225–232 (1997). https://doi.org/10.1145/258734.258854

[15] Hoiem, D., Efros, A.A., Hebert, M.: Automatic photo pop-up. In: ACM SIGGRAPH 2005 Papers. SIGGRAPH '05, pp. 577–584. Association for Computing Machinery, New York, NY, USA (2005). https://doi.org/10.1145/1186822.1073232 . https://doi.org/10.1145/1186822.1073232

[16] Saxena, A., Sun, M., Ng, A.Y.: Make3D: Learning 3D Scene Structure from a Single Still Image. IEEE Transactions on Pattern Analysis and Machine Intelligence **31**(5), 824–840 (2009) https://doi.org/10.1109/TPAMI.2008.132

[17] Niklaus, S., Mai, L., Yang, J., Liu, F.: 3d ken burns effect from a single image. ACM Transactions on Graphics (ToG) **38**(6), 1–15 (2019)

[18] Wiles, O., Gkioxari, G., Szeliski, R., Johnson, J.: Synsin: End-to-end view synthesis from a single image. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 7467–7477 (2020)

[19] Hedman, P., Kopf, J.: Instant 3d photography. ACM Transactions on Graphics (TOG) **37**(4), 1–12 (2018)

[20] Tucker, R., Snavely, N.: Single-view view synthesis with multiplane images. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 551–560 (2020)

[21] Dickson, A., Knott, A., Zollmann, S.: User-centred Depth Estimation Benchmarking for VR Content Creation from Single Images. In: Lee, S.-H., Zollmann, S., Okabe, M., Wünsche, B. (eds.) Pacific Graphics Short Papers, Posters, and Work-in-Progress Papers. The Eurographics Association, ??? (2021). https://doi.org/10.2312/pg.20211394

[22] Waidhofer, J., Gadgil, R., Dickson, A., Zollmann, S., Ventura, J.: Panosynthvr: Toward light-weight 360-degree view synthesis from a single panoramic input. In: 2022 IEEE International Symposium on Mixed and Augmented Reality (ISMAR), pp. 584–592 (2022). https://doi.org/10.1109/ISMAR55827.2022.00075

[23] Wang, Q., Li, Z., Salesin, D., Snavely, N., Curless, B., Kontkanen, J.: 3D Moments from Near-Duplicate Photos. arXiv (2022). https://doi.org/10.48550/ARXIV.2205.06255 . https://arxiv.org/abs/2205.06255

[24] Barron, J.T., Mildenhall, B., Tancik, M., Hedman, P., Martin-Brualla, R., Srinivasan, P.P.: Mip-nerf: A multiscale representation for anti-aliasing neural radiance fields. In: Proceedings of the IEEE/CVF International Conference on Computer Vision, pp. 5855–5864 (2021)

[25] Martin-Brualla, R., Radwan, N., Sajjadi, M.S., Barron, J.T., Dosovitskiy, A., Duckworth, D.: Nerf in the wild: Neural radiance fields for unconstrained photo collections. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 7210–7219 (2021)

[26] Yu, A., Ye, V., Tancik, M., Kanazawa, A.: pixelnerf: Neural radiance fields from one or few images. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 4578–4587 (2021)

[27] Newcombe, R.A., Izadi, S., Hilliges, O., Molyneaux, D., Kim, D., Davison, A.J., Kohi, P., Shotton, J., Hodges, S., Fitzgibbon, A.: Kinectfusion: Real-time dense surface mapping and tracking. In: 2011 10th IEEE International Symposium on Mixed and Augmented Reality, pp. 127–136 (2011). IEEE

[28] Müller, T., Evans, A., Schied, C., Keller, A.: Instant neural graphics primitives with a multiresolution hash encoding. ACM Trans. Graph. **41**(4), 102–110215 (2022) https://doi.org/10.1145/3528223.3530127

[29] Li, T., Slavcheva, M., Zollhoefer, M., Green, S., Lassner, C., Kim, C., Schmidt, T., Lovegrove, S., Goesele, M., Lv, Z.: Neural 3d video synthesis. arXiv preprint arXiv:2103.02597 (2021)

[30] Xian, W., Huang, J.-B., Kopf, J., Kim, C.: Space-time neural irradiance fields for free-viewpoint video. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pp. 9421–9431 (2021)

[31] Pumarola, A., Corona, E., Pons-Moll, G., Moreno-Noguer, F.: D-nerf: Neural radiance fields for dynamic scenes. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 10318–10327 (2021)

[32] Park, K., Sinha, U., Hedman, P., Barron, J.T., Bouaziz, S., Goldman, D.B., Martin-Brualla, R., Seitz, S.M.: Hypernerf: A higher-dimensional representation for topologically varying neural radiance fields. arXiv preprint arXiv:2106.13228 (2021)

[33] Song, L., Chen, A., Li, Z., Chen, Z., Chen, L., Yuan, J., Xu, Y., Geiger, A.: Nerfplayer: A streamable dynamic scene representation with decomposed neural radiance fields. IEEE Transactions on Visualization and Computer Graphics **29**(5), 2732–2742 (2023)

[34] Fang, J., Yi, T., Wang, X., Xie, L., Zhang, X., Liu, W., Nießner, M., Tian, Q.: Fast dynamic radiance fields with time-aware neural voxels. In: SIGGRAPH Asia 2022 Conference Papers, pp. 1–9 (2022)

[35] Yang, Z., Gao, X., Zhou, W., Jiao, S., Zhang, Y., Jin, X.: Deformable 3d gaussians for high-fidelity monocular dynamic scene reconstruction. arXiv preprint arXiv:2309.13101 (2023)

[36] Rematas, K., Kemelmacher-Shlizerman, I., Curless, B., Seitz, S.: Soccer on your tabletop. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 4738–4747 (2018)

[37] Chen, J., Paris, S., Wang, J., Matusik, W., Cohen, M., Durand, F.: The video mesh: A data structure for image-based three-dimensional video editing. In: 2011 IEEE International Conference on Computational Photography (ICCP), pp. 1–8 (2011). https://doi.org/10.1109/ICCPHOT.2011.5753118

[38] Zollmann, S., Dickson, A., Ventura, J.: Casualvrvideos: Vr videos from casual stationary videos. In: 26th ACM Symposium on Virtual Reality Software and Technology, pp. 1–3 (2020)

[39] Hwang, D.-H., Koike, H.: Parapara: Synthesizing pseudo-2.5d content from monocular videos for mixed reality. In: Extended Abstracts of the 2018 CHI Conference on Human Factors in Computing Systems. CHI EA '18, pp. 1–6. Association for Computing Machinery, New York, NY, USA (2018). https://doi.org/10.1145/3170427.3188596 . https://doi.org/10.1145/3170427.3188596

[40] Sturm, J., Engelhard, N., Endres, F., Burgard, W., Cremers, D.: A benchmark for the evaluation of rgb-d slam systems. In: Proc. of the International Conference on Intelligent Robot Systems (IROS) (2012)

[41] Ranftl, R., Bochkovskiy, A., Koltun, V.: Vision transformers for dense prediction. In: Proceedings of the IEEE/CVF International Conference on Computer Vision, pp. 12179–12188 (2021)

[42] Nathan Silberman, P.K. Derek Hoiem, Fergus, R.: Indoor segmentation and support inference from rgbd images. In: ECCV (2012)

[43] Wu, Y., Kirillov, A., Massa, F., Lo, W.-Y., Girshick, R.: Detectron2. https://github.com/facebookresearch/detectron2 (2019)

[44] Schönberger, J.L., Frahm, J.-M.: Structure-from-Motion Revisited. In: Conference on Computer Vision and Pattern Recognition (CVPR) (2016)

[45] Zeng, A., Song, S., Nießner, M., Fisher, M., Xiao, J., Funkhouser, T.: 3dmatch: Learning local geometric descriptors from rgb-d reconstructions. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 1802–1811 (2017)

[46] Newcombe, R.A., Fox, D., Seitz, S.M.: Dynamicfusion: Reconstruction and tracking of non-rigid scenes in real-time. In: 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 343–352 (2015). https://doi.org/10.1109/CVPR.2015.7298631

[47] Suvorov, R., Logacheva, E., Mashikhin, A., Remizova, A., Ashukha, A., Silvestrov, A., Kong, N., Goka, H., Park, K., Lempitsky, V.: Resolution-robust large mask inpainting with fourier convolutions. arXiv preprint arXiv:2109.07161 (2021)

[48] Telea, A.: An image inpainting technique based on the fast marching method. J. Graphics, GPU, & Game Tools **9**(1), 23–34 (2004)

[49] Lorensen, W.E., Cline, H.E.: Marching cubes: A high resolution 3d surface construction algorithm. In: Seminal Graphics: Pioneering Efforts that Shaped the Field, pp. 347–353 (1998)

[50] Zhou, T., Brown, M., Snavely, N., Lowe, D.G.: Unsupervised learning of depth and ego-motion from video. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 1851–1858 (2017)

[51] Zhang, C.: Investigation on motion sickness in virtual reality environment from the perspective of user experience. In: 2020 IEEE 3rd International Conference on Information Systems and Computer Aided Education (ICISCAE), pp. 393–396 (2020). IEEE

[52] Guédon, A., Lepetit, V.: Sugar: Surface-aligned gaussian splatting for efficient 3d mesh reconstruction and high-quality mesh rendering. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 5354–5363 (2024)

[53] Liu, Q., Liu, Y., Wang, J., Lv, X., Wang, P., Wang, W., Hou, J.: Modgs: Dynamic gaussian splatting from causually-captured monocular videos. arXiv preprint arXiv:2406.00434 (2024)

[54] Dai, A., Nießner, M., Zollhöfer, M., Izadi, S., Theobalt, C.: Bundlefusion: Real-time globally consistent 3d reconstruction using on-the-fly surface reintegration. ACM Transactions on Graphics (ToG) **36**(4), 1 (2017)

[55] Schrepp, M.: User Experience Questionnaire Handbook-Version 8. URL: https://www.ueq-online.org/Material/Handbook.pdf (2019)

[56] Regenbrecht, H., Schubert, T.: Real and illusory interactions enhance presence in virtual environments. Presence **11**(4), 425–434 (2002)

[57] Schubert, T., Friedmann, F., Regenbrecht, H.: The experience of presence: Factor analytic insights. Presence: Teleoperators & Virtual Environments **10**(3), 266–281 (2001)

[58] Schwind, V., Knierim, P., Haas, N., Henze, N.: Using presence questionnaires in virtual reality. In: Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems, pp. 1–12. Association for Computing Machinery, New York, NY, USA (2019). https://doi.org/10.1145/3290605.3300590

[59] Baker, L., Ventura, J., Zollmann, S., Mills, S., Langlotz, T.: SPLAT: Spherical Localization and Tracking in Large Spaces. In: IEEE Virtual Reality (IEEE VR) (2020)

[60] Baker, L., Mills, S., Zollmann, S., Ventura, J.: CasualStereo: Casual Capture of Stereo Panoramas with Spherical Structure-from-Motion. In: Proceedings - 2020 IEEE Conference on Virtual Reality and 3D User Interfaces, VR 2020 (2020). https://doi.org/10.1109/VR46266.2020.1581313146787