2025 IEEE Conference on Virtual Reality and 3D User Interfaces Abstracts and Workshops (VRW) | 979-8-3315-1484-6/25/$31.00 ©2025 IEEE | DOI: 10.1109/VRW66409.2025.00086

3D Depth Experience in VR from Monocular Video

Christoph Nickas[*] Graz University of Technology
Philipp Fleck[†] Graz University of Technology
Thomas Kernbauer[‡] Graz University of Technology
Clemens Arth[§] Graz University of Technology

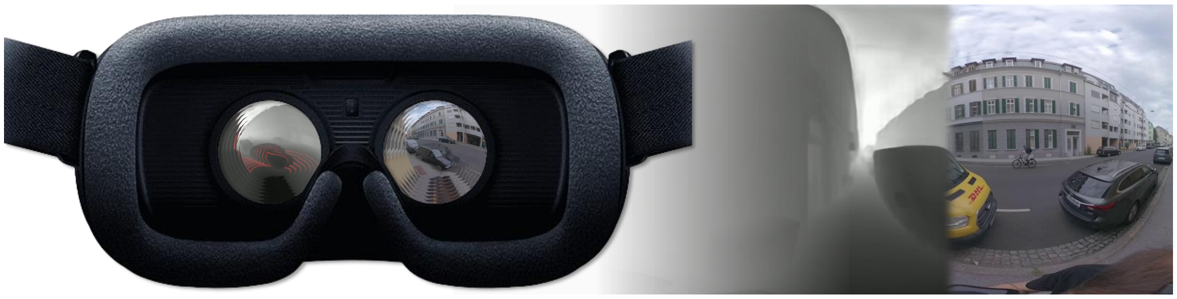
Figure 1: Image showing the basic concept of our algorithm. On the left, a VR headset illustrating the rendering of multiple spheres is depicted. On the right, a depth image and the corresponding real image is shown.

Abstract

Virtual Reality (VR) lives from the experience of diving into a world which is perceived as a three-dimensional environment. Regular two-dimensional video materials captured in the real world are only partially suitable to convey the real feeling of presence in VR. This paper describes the design and implementation of a 3D effect on a video in VR through the use of only a single monocular video as input. This is realized through a monocular depth estimation algorithm, which pre-processes the video and extracts depth information, creating a corresponding depth map video. An application running on a VR headset uses this depth map to create the 3D effect through projecting the video onto multiple spheres at different sizes and distances from the viewer, depending on the corresponding depth value. We describe the entire process and show a few examples, analyzing the individual algorithmic steps.

Index Terms: Artificial Intelligence, Depth Estimation, Monocular Video, Virtual Reality

1 Introduction

Virtual Reality has gained more and more traction in recent years. The main focus is often to simulate a fictional scene to dive into. As technology evolved over time and the capture of high quality images and videos of the real world is more accessible than ever, the demand for photorealistic content for VR increases in popularity. Through the introduction of affordable 360° cameras, the market became saturated with surround images and videos with a clear application to VR. The possibility of immersing yourself in real places and situations as if you were actually present, or traveling the world without ever leaving the room is incredible. But what if we could improve on that to make it even more immersive and realistic?

We, as humans, do not visually perceive our surrounding environment through optical vision alone. Depth perception allows us to experience the world around us as a three-dimensional space instead of just a two-dimensional plane. This works through the use of our two eyes, which have enough difference in viewpoint to allow our brain, with the help of contextual clues, to assess distance. Revisiting 360° video footage shot by a consumer camera, we quickly realize that it is flat, similar to every other ordinary camera; the only difference being that it is spherical instead of planar, which allows us to look in different directions. When using a VR headset, we can decide the content that each individual eye can see. Therefore, using two separate images with a slightly different viewpoint and using them for both the left and the right eyes, respectively, the brain is tricked into thinking that it is seeing depth. However, we do not have the information to properly separate viewpoints from monocular images or videos.

Suppose we had depth information for 360° video footage. Many applications would benefit, such as tourism, which relies on experience-driven activities like hiking and scuba diving that aim to evoke emotions. Advertising these is hard, as VR headsets with 360° images often miss accurately conveying height sensations. Adding depth would create a more realistic experience and attract more participants. The entertainment industry seeks to captivate through sight and sound, but traditional screens and 2D VR lack depth perception. Enhanced 360° videos would improve experiences, especially in dynamic activities like skiing. Moreover, it would allow those with limited mobility to virtually explore hard-to-reach places.

Creating effective video footage usually requires a deliberate process each time an image or video is captured, making previous images unsuitable for this effect. We present a method for applying depth enhancement easily to existing 360° images, enhancing immersive content. Our system uses a monocular depth estimation method [29] to derive depth from contextual clues, allowing us to represent these values as actual depth in VR. Handling the large amount of depth data in videos is challenging, so we explore different representation methods. Consequently, the depth effect functions on a state-of-the-art VR headset with real-time processing, excluding depth calculation.

The rest of the current document is structured as follows. In Section 2 we revisit different related works in the area of depth estimation and devices. Section 3 is dedicated to the description of our algorithmic pipeline. In Section 4 we give details about the practical implementation, while we show some experimental results and analysis in Section 5. Finally, concluding remarks are given in Section 6.

[*]e-mail: christoph.nickas@student.tugraz.at
[†]e-mail: philipp.fleck@tugraz.at
[‡]e-mail: kernbauer@tugraz.at
[§]e-mail: clemens.arth@tugraz.at

2 Related Work

Photography and video creation for information or entertainment is widespread and familiar. Distance information *i.e.,* indicating the separation between two points, can be measured in the real world or its 3D digital model. Depth abstracts this such that each pixel is linked to a distance from the camera to a 3D point on an object in reality or its digital model. Creating this depth information is essential to enhance 2D video immersion to 3D in VR, which we aim to address.

In the following, we first discuss representation methods, followed by an investigation of ways to measure or estimate distances and depth, respectively, which can serve as a supplement to processing images. Although there is a huge body of literature, we mainly focus on works related to 360° imagery and suitable capturing devices, VR and the findings and shortcomings.

2.1 Environment Representation Methods

Wang [28] provides a review of different 3D representation methods. We discuss several models and also discuss 2D image-based representation methods as follows.

Point Clouds and Meshes Point clouds are a general key method for representing environments and are closely linked to Light Detection and Ranging (LiDAR), as Leberl *et al*. [15] discusses. They depict objects through a set of points with each point specified by three coordinates (x,y,z), and can include additional data such as *e.g.,* color to enhance visualization. Point clouds offer raw 3D storage due to their high resolution and editing flexibility.

A deferred representation involves 3D meshes and rendering techniques, as discussed *e.g.,* by Kato *et al*. [10]. A 3D mesh, made of vertices and edges, forms a texturable surface. It allows point reduction without creating surface holes, enhancing performance and reducing file size. However, it requires additional computation for mesh grid and reduction.

Ueda *et al*. [24] uses light-field imagery and proposes the use of multiple planes at varying depths to simplify scene division. Broxton *et al*. [4] show how this idea can be used for 360° imagery, substituting planes with spheres of different radii.

Spherical Image Representation Capturing a 360-degree video involves recording multiple videos from the same viewpoint and focal length, with varying directions. This rotational combination creates a spherical or surround video feed. Since we cannot look in all directions simultaneously, the video is often projected onto a sphere's interior, as per *e.g.,* Liu *et al*. [17], enabling a camera to select any view direction, thus offering an immersive experience. Gkitsas *et al*. [6] present PanoDR, a system for Diminished Reality (DR) using spherical imagery. Mühlhausen *et al*. [20] present a learning-based system also using multi-sphere images to enable full 6 degree-of-freedom (DoF) head motion.

A key issue is the absence of a distinct spherical video format due to compatibility concerns. To create a planar video, spherical recordings are projected onto a plane and can be reverted back using inverse projection. Kennedy *et al*. [12] describes several methods for spherical-to-planar transitions, with equirectangular projection being the most common.

Image-based Rendering (IBR) Both Neural Rendering Fields (NERFs) and Gaussian Splatting are novel view synthesis methods, aiming to render scenes, typically captured in the real world, entirely based on images. In a nutshell, a novel view is generated from the content of a multitude of existing images of a scene. An overview of the field is given by Shum and Kang [23], while we abstain from a more in-depth review of this very active field of research. Attal *et al*. [2] present MatryODShka for novel view synthesis, which uses stereo 360° imagery and a multilayered spherical representation. The exemplary work of Jiang *et al*. [9] recently impressively demonstrated interactivity. It is important to note that these methods are still hard to apply practically on conventional headsets; scenes need to be captured from multiple view points, preprocessing and rendering requires significant amounts of computational and memory resources. However, IBR is considered a key future method for reproducing photorealistic content in VR.

2.2 Estimation of Depth

The estimation of depth and distances is a broad field of research, which has been boosted recently particularly through the development of new hardware technologies and deep neural networks.

Lasers: A common approach to depth data collection is the use of laser sensors. Most prominently, LiDAR can be used to obtain the distance to certain points of a scene. This method is also used by the possibly most famous 360° depiction of real places, Google Street View, described by Anguelov *et al*. [1]. The majority of images and depth information captures for their database come from a highly specialized and costly hardware setup containing a multitude of cameras and laser sensors mounted on top of a moving vehicle. The exact structure has changed over the years, but the main concept has remained the same. A spherically arranged array of cameras is periodically taking pictures of every possible direction. During driving, this results in a collection of pictures every few meters, which are merged into a single spherical image. Kim *et al*. [13] are using a similar setup to create a 360° dataset including images and LiDAR scans. Arguably, laser-assisted 360° capturing setups are costly and complicated to operate.

Stereo Matching: A more traditional, yet well established approach to gathering depth is stereo depth estimation [21, 7], more recently also incorporating learning frameworks such as the work of Martins *et al*. [18]. This method employs the differences among several cameras placed at slightly varied angles to triangulate distances. However, using two 360° cameras instead of two conventional cameras leads to the creation of a handful of new problems caused by the 360° field of view, described by Matzen *et al*. [19]. Mutual obstruction or the varying strength of disparity between the cameras depending on the viewing direction is a general problem, which they overcome by creating a specialized construction: two 360° cameras with a fixed offset, constantly spinning both around themselves and their common axis of rotation. This allows them to successfully stereo match both video streams in order to gather depth information. Broxton *et al*. [4] also describe a custom capturing rig, which uses 46 individual cameras to create a 360° image, including depth information through stereo matching.

Depth from Motion: To estimate depth using a single monocular camera, the camera must undergo some motion between successive frames. The idea is to compare two of these frames and analyze the change between them, which is conceptually similar to stereo depth estimation, just without information about the relative camera position, which makes it less accurate. One such method is depth through optical flow paired with deep neural networks, proposed by Wang *et al*. [27]. Although this works well for pinhole cameras, the biggest limitation is created by the nature of 360° imagery. Depth information perpendicular to the moving direction can be extracted properly, as there is usually enough baseline. However, the computation of the optical flow in the moving direction often exposes insufficient difference between frames to accurately compute depth, thus the depth map lacks detail.

Depth from Context: More recently, the estimation of depth from contextual clues with the help of deep neural networks has become very popular. In such an approach, a model is trained on both labeled and unlabeled datasets while finally being able to predict the depth from new, yet unseen, image data. Bhat *et al*. [3] proposed ZoeDepth, which is considered the current state-of-the-art in that field and is based on MiDAS, an earlier work by Ranftl *et al*. [22]. Yang *et al*. [29] proposed Depth Anything, which follows a similar approach and provides an improvement with respect to the metric depth values. Ke *et al*. [11] proposed Marigold, while Li *et al*. [16] proposed Patchfusion, which are both capable of achieving more accurate results with the drawback of a large increase in computation time.

2.3 Spherical Cameras and Scanners

Even professional 360° video equipment struggles to fulfill the requirements for high framerates and resolutions in VR. A wide range of different 360° cameras are available on the consumer market, varying considerably in price and affordability. The most prominent examples include the Insta360 X series, the Ricoh Theta, the GoPro MAX and the Samsung Gear 360 class of devices. These devices clearly target the consumer market for more conventional 360° image and video acquisition. While those cameras are comparably cheap, the video resolution is restricted around the 2-4k range, while frame rates are limited to 30-60fps at max.

More expensive devices dedicated to professional VR content creation include the Insta360 Titan, the KanDao QooCam 8K or the Teledyne Ladybug or Mosaic spherical cameras. Those devices are hardly affordable; however, they can provide 8k video at proper framerates up to 120fps and usually serve as capturing devices in professional video broadcasting. This class of devices is also considered to provide adequate content for state-of-the-art professional video replay in VR today.

3 Depth-Enhanced 360° Video for VR

The basic input to our approach is a single 360° video captured with a monocular camera, in which we do not place any requirements on the actual image resolution or frame rate. However, the higher the resolution and frame rate of the input video, the better the final experience in VR.

In Figure 2, a high-level overview of our approach is shown, separated into two basic stages, a video pre-processing pipeline and live video replay. During pre-processing, we extract each individual frame from the video and estimate the depth using a monocular depth estimation method. The depth map is encoded into a new accompanying video. During playback, a set of differently sized concentric spheres is created, which are assembled as the shells of an onion. Thresholds on the calculated depth define a masking for each individual sphere, giving the final multi-spherical representation. In the VR experience, the left and right eyes are rendered from the inside of the sphere from two displaced cameras with different viewports. Note that we postpone a discussion of the impact of individual parameters mentioned throughout the rest of this Section until the discussion of our evaluation results in Section 5.

3.1 Depth from Video

At the core of our method lies a contextual deep learning approach for the extraction of depth values, namely Depth Anything by Yang *et al*. [29]. We built a framework around it to streamline the processing of videos, which includes everything necessary to go from an ordinary 360° video to a combined video consisting of the said video and its respective depth map.

Frame Processing First, we extract each individual frame from the input video to forward them to the depth estimation stage.

Depth Anything is not particularly designed to handle 360° panoramic imagery. This creates certain problems, which originally stem from the imagery used to train the deep learning model. Physical lens assemblies cause a brightness fall-off towards the borders of an image, as described by Unger *et al*. [25]. In digital photography, this effect is usually compensated for at least partially by digital amplification, and it is almost unnoticeable to the human eye. Still, the depth estimation at the borders suffers from inaccuracies exactly because of this effect, which is less harmful to pinhole camera images. For 360° panoramas, this is an issue, as the panoramic image indeed "does not end" at the left and right borders.

Another problem comes from the basic nature of the deep learning framework itself. Because Depth Anything is a system estimating depth from visual clues, again, at the borders the amount of information is limited. Although one would assume that the estimated depth at the left and right borders of the image is almost the same, *i.e.,* the depth estimation function across the borders is continuous, this is actually not the case. We counteract both effects by artificially enlarging the panoramic images at the borders. In Figure 3, the extension methodology is shown; overall, it is a plain copy of a set of *n* pixel columns from left to right and vice versa.

Computing the Depth Map Based on the padded imagery, we aim to calculate a depth map which should be consistent in the time domain, as we are looking at videos, rather than single images. This means that the depth values for individual pixels calculated for successive frames should be similar, unless there is a sudden change caused by a real depth discontinuity, such as *e.g.,* the corner of a building. In order to achieve this, we use a particular Depth Anything model, which is trained to deliver metric depth values. This contrasts to the default usage of the deep neural network that delivers relative depth values, *i.e.,* values that are scaled between a defined maximum and minimum value. Using metric values is essential to retrieve depth values for individual pixels that are approximately the same for successive frames, unless the pixel content indeed changes drastically.

In addition to offering models trained on indoor and outdoor scenes, Depth Anything can also be provided with a polling size to scale computational demands and accuracy. In other words, it is possible to provide a lower-resolution version of an image to the algorithm rather than a full-resolution frame. A lower polling size decreases computational demands but also decreases the accuracy and resolution of the resulting depth array, the default outcome of the algorithm, at the same time. Regardless of the polling size, the array is adequately resized to fit the video aspect ratio, considering a remaining aspect of the algorithm: the precision of the estimated depth values. In order to store the resulting depth values efficiently in image space, the depth values are converted into a simple 8-bit format, ranging from 0, being closest to the camera, to 255 being farthest from the camera. Note that at this point, we crop the same number of *n* columns on both sides of the depth map to fit the original video resolution, or the respective fraction of *n* columns, if the polling size and resulting depth image are smaller than the original frame size.

Two examples of a video frame and its corresponding depth map are shown in Figure 4. Iterating over all frames of a video results in a second set of corresponding grayscale frames encoding the respective depth maps. These grayscale frames can be encoded again into a video file using proper video encoding algorithms.

3.2 Recreating the Depth Effect in VR

To recreate the aimed depth effect in VR, a dedicated set-up of concentric spheres and proper algorithms for texturing these spheres are required. We thereby assume that at the end, the observer in VR is placed in the center of this multi-spherical setup, similar to the default way of creating experiences based on 360° panoramic videos in VR using a single surrounding spherical surface.

Spherical Representation In order to texture a single sphere from an image or video frame, we use a proper UV mapping. We convert the horizontal and vertical axes of the equirectangular video frame (u,v) into the corresponding coordinates of the sphere's surface (x,y,z) by two formulas in Equations 1. Each individual point on the surface of the sphere is given by a unit vector (d_x,d_y,d_z) with

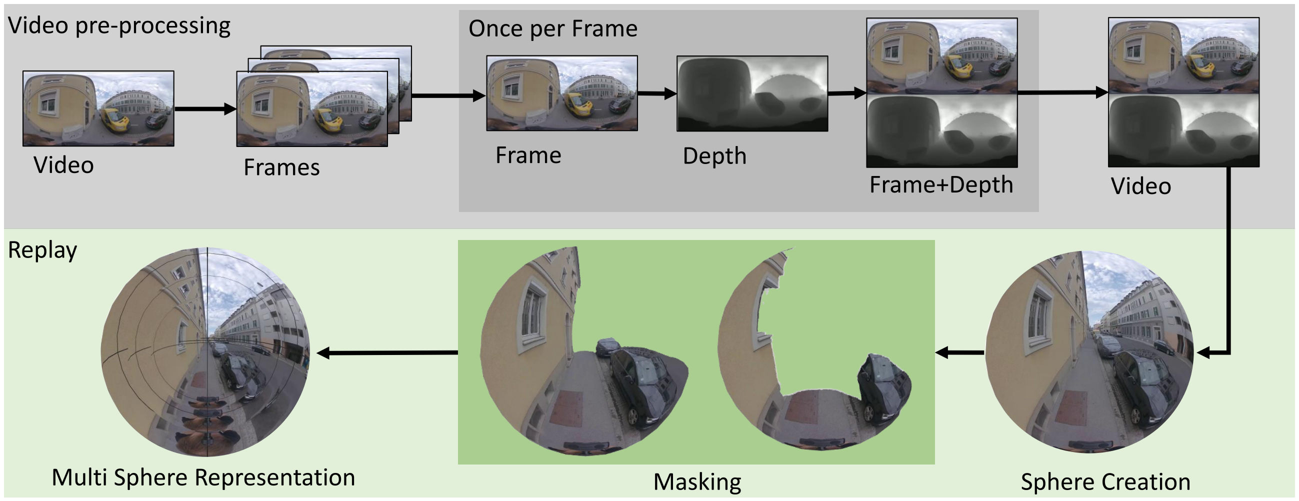

Figure 2: An overview of our approach. First, we extract each individual frame from the video and estimate a depth map. Both the frame and the depth map are combined into a new video. During playback, masks based on depth thresholds are applied to differently sized concentric spheres. Placing the left and right eye accordingly close to the center of the spheres gives the final experience.

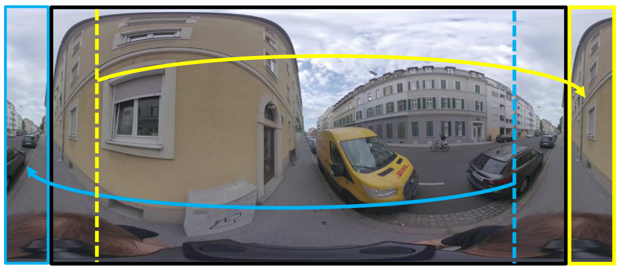

Figure 3: Visualization of the border extension: The blue and yellow areas are symbolizing exactly which parts of the frame are copied.

its origin at the center of the sphere.

$$u = 0.5 + \frac{\operatorname{arctan2}(d_z, d_x)}{2\pi}, \quad v = 0.5 + \frac{\arcsin(d_y)}{\pi}. \tag{1}$$

However, a single sphere cannot convey any sort of depth information even when scaling the sphere radius, except for varying the amount of potential eye strain when viewed in VR, as studied by Coles *et al.* [5].

Our setup uses multiple spheres at different distances, placing different parts of a scene on their respective spheres based on the depth, *i.e.,* distance, estimate. Because we use two slightly off-center cameras in VR, when focusing on a fixed point in a scene, each eye creates an imaginary line called the visual axis. Together, both lines form a vergence angle in between them. Depending on the distance to the point, this angle also changes. The farther away it is, the angle becomes more acute, as shown exemplarily on the left of Figure 5 with angles α and β. Vergence is one of the aspects that allow the human brain to estimate depth and distance.

The other big factor is the binocular disparity. It describes the slight difference between the images captured by the eyes, created by the horizontal offset of their viewpoints. This difference increases the closer an object is to the observer and is measurable by comparing the left- and right-eye view of a scene. As shown exemplarily on the right of Figure 5, the view is focused on a given point F with an obstructing point O in the foreground. The binocular disparity is measurable through the difference of the two resulting angles, *i.e.,* γ and δ, in between the line of sight of each eye. The human brain also uses this dissimilarity in order to perceive depth.

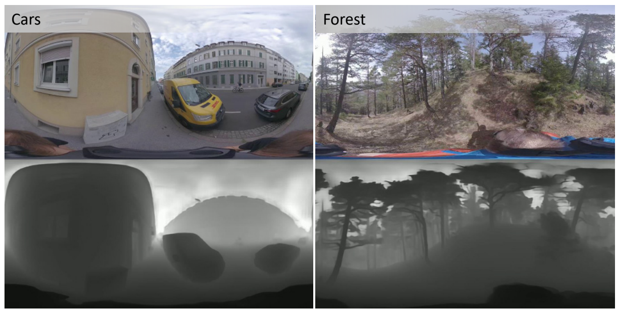

Figure 4: An example of how resulting frames would look like after the preprocessing is complete. (**Cars**) An urban scene showing multiple cars and buildings. **(Forest)** A natural scene showing a multitude of trees inside a montane forest.

The concentric spheres have radii that are linearly correlated to the expected metric depth values of the imagery. In order to choose the radii properly, both the vergence angle and the binocular disparity between the spheres should remain as small as possible, to avoid increased eye strain and the creation of irritating pattern repetitions at depth discontinuities. This essentially defines the strategy conceptually: using as many and as closely spaced spheres as possible.

However, having as many spheres as depth levels is usually impractical for performance reasons, as also mentioned by Wang *et al.* [26] and Ishikawa *et al.* [8] in a related context. Moreover, the depth estimates are approximated by the neural network, rather than being true values. For each sphere, the image content is therefore defined by the respective depth values within a given range centered around the respective sphere radius. One notable benefit of this method is its flexibility in addressing both detail level and performance, while still being able to take the nature of the actual scenario into account. In other words, the spacing of spheres can be adjusted to indoor and outdoor scenarios, such that the respective experience in VR is optimized.

4 Implementation

In the following, we discuss certain aspects of a practical implementation of our depth effect. Note that we focus on the use of conventional hardware and software and on the viability for enthusiasts, abstaining from the discussion of special video setups or user customization.

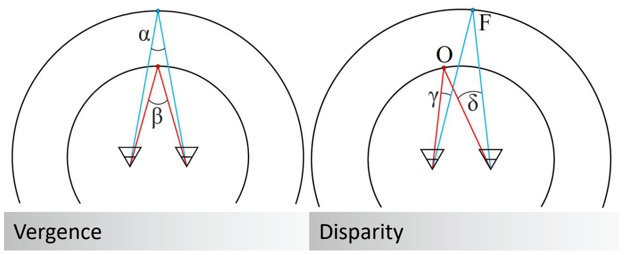

Figure 5: **(Vergence)** Vergence Angle: The magnitude of the angle, *i.e.,* α and β, decreases as the distance increases. **(Disparity)** Binocular Disparity: From the perspective of the left camera, points F and O appear much closer together than they do from the right camera, measurable through the angles γ and δ.

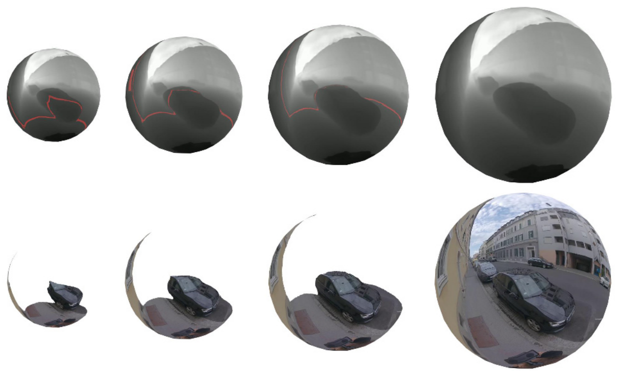

Figure 6: Outside view of individual spheres at different sizes with their corresponding masking: The top row images are a view of the corresponding depth map with the edges of each sphere marked in red for better visibility.

4.1 Video Preprocessing

Each video has to be preprocessed to extract the depth information. Once all raw depth images are available, the most obvious approach is to encode the frames into a second separate video. However, this causes several problems during playback later on, as both videos need to be played back simultaneously in total sync. Therefore, we create a single video that combines the original frame and the depth map. An important consideration here is that there are limits to the format of state-of-the-art video encoders and decoders. Particular resolution limits apply to older H.264 codec[1] profiles, while for newer levels hardware acceleration is not supported on all devices; therefore, the H.265 codec[2] is usually used, while the maximum video width and height are still limited to 8192×4320 pixels[3]. Moreover, video formats do not contain alpha channels, which we could use to supplement the original imagery with a fourth channel.

We can align the depth frames right below or beside the RGB image for resolutions of 4k and slightly above, staying within resolution bounds of H.265 and requiring only one player instance at runtime. However, if a full-resolution depth video is desired for higher resolutions, this has to be stored in a separate video file, or a lower-resolution version of the depth frame has to be used, trading convenience against level of detail.

4.2 Runtime Application

Most VR applications are built on the Unity game engine [14], so is our prototypical implementation. Because we want to leverage the flexibility of adapting the number and radii of spheres, only a

[1]MPEG-4, Advanced Video Coding (Part 10) (H.264)

[2]High Efficiency Video Coding (HEVC) Family, H.265, MPEG-H Part 2

[3]We refer to hardware limitations on mobile devices, such as the Quest3.

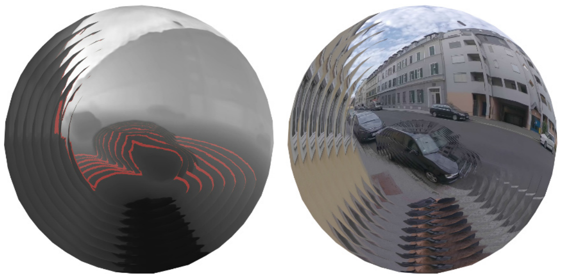

Figure 7: Outside view of a combined, texture mapped and masked multi sphere setup: On the left is a view of the corresponding depth map, with the edges of each sphere marked in red for better visibility.

single sphere is created initially within the default scene. At the center of the sphere is a rigid set of two cameras, linked to each of the two displays inside the VR headset. Both cameras face the same direction with a constant baseline equal to the average distance between the eyes of a user (*i.e.,* approximately 67 mm). The gyroscopic sensor of the VR headset is used to properly adapt the rotation and translation of the camera rig about its center, as the user's head moves.

To display a single image as a set of depth-separated spheres, we use a fragment shader, also known as a pixel shader, implementing the two UV mapping in Equation 1. The shader maps the entire image texture to the outside of the sphere, and we use front-face culling to create an unobstructed view onto the sphere from the inside[4]. Based on the given scenario to display, we replicate the textured sphere with the same center multiple times with different radii, which results in an onion-shaped set of fully textured spheres.

A material shader is used for masking each sphere depending on the depth map. For each individual pixel, we look up its corresponding depth value and compare it to the desired range of the spheres. Setting the corresponding value in the alpha channel to its extremes, *i.e.,* 1 or 0 for opaque and transparent, respectively, creates very harsh edges at depth discontinuities, *i.e.,* discontinuities of the respective sphere contents. We therefore counteract this effect by using a gradual alpha blending around the edges, which is essentially a linear transition from 0 to 1 over a range of a few pixels. On the one hand, this creates a loss in sharpness; however, at the same time it partially obscures the limit in the number of spheres used. In Figure 6 the masked spheres and the individual depth boundaries are shown. Figure 7 depicts a flattened version of the combination of all individual spheres.

For the replay of a video, this process is essentially repeated for each video frame delivered by a player instance. Due to the limits of conventional VR headsets in terms of computational and memory resources, the number of spheres has to be clearly limited to the range of 10-20, also depending on the video resolution and the actual video frame rate. Note that the lookup of the individual values from the depth map is essentially only a reference, *i.e.,* an offset in UV coordinates, to a different portion of the decoded video frame, in case the depth frame has been directly encoded in the video.

5 Evaluation

To collect our test scenes, we used an Insta360 X3 camera, which is a commonly used consumer camera. The camera was held in hand most of the time and in some instances mounted to a backpack. The Insta360 cameras are not directly saving the recordings in a highly compatible video format, but use a proprietary format. The official

[4]Note that we resort to using standard options in Unity here, while more efficient ways to achieve this goal certainly exist.

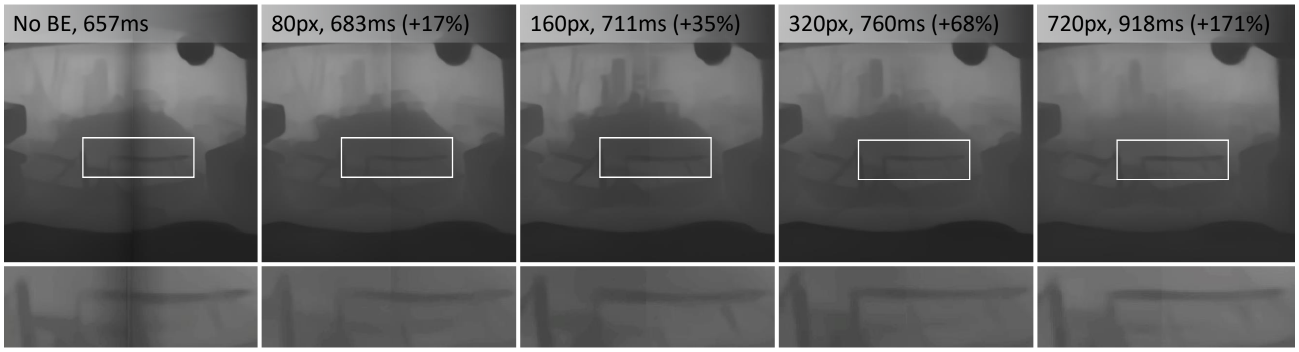

Figure 8: Comparison between different border extension widths based on a resolution of 1440x720 pixels. The first row shows the used extension width in pixels and the computation time for the depth map in absolute and relative terms, together with the resulting depth frame. The second row shows a cropped image of how the seam between the two borders changes due to the border extension process.

	Cars	Barn	Forest	Lecture
Resolution [px]	5760×2880	5760×2880	5760×2880	5760×2880
Frame rate [FPS]	30	30	30	30
Duration [min]	01:07	01:52	96:14	02:06
File size [GB]	0.54	0.84	58.7	0.76
Walking distance	60 m	50 m	3 km	100 m

Table 1: A detailed overview of some important parameters of different scenes.

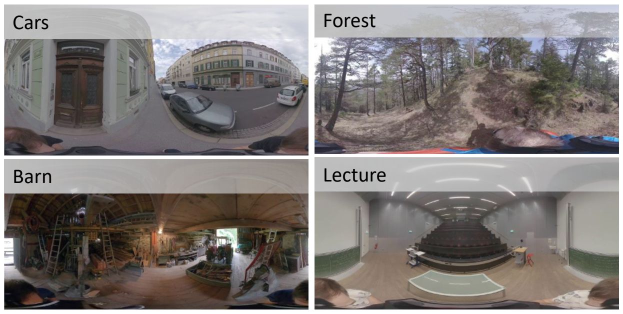

Figure 9: Exemplary frames from our four datasets: Cars, Barn, Forest, Lecture.

Insta360 Studio software must be used to export and convert the video to an MP4/H.265 file format.

We recorded at 5.7K resolution, which is the highest available and equivalent to a resolution of 5760×2880 pixels, while the framerate is limited to 30 fps. The corresponding video bitrate averages 70 MBit/s. In Table 1 an overview of the most important parameters is provided. All preprocessing was performed on a conventional desktop computer with an Nvidia RTX 4070 graphics card.

5.1 Border Extension Effects

As mentioned in Section 3.1, we need to pad the original images on the side replicating image data to counteract certain effects in the depth estimation pipeline. To illustrate the importance of this measure, we conducted experiments with different padding widths, also recording the induced computational overhead.

In our example, the downscaled frame has a size of 1440×720 pixels. In Figure 8 the results of different border widths are shown, including a close-up of the depth values at the image seam in the second row. As expected, the discontinuity at the seam becomes more and more unnoticeable as the border size increases, while the computational efforts increase at the same time. An extension of 720 pixels on each side essentially results in double the total number of pixels processed by the depth estimation engine, as shown in the far left column in Figure 8. The seem appears to vanish almost entirely in the middle column, corresponding to a padding of 160 pixels at each side of the image. This indicates a good trade-off between the induced costs and the depth accuracy to settle at a padding size which is equivalent to about 20% of the original image, *i.e.,* about 10% padded to each image side.

5.2 Scene Analysis

Several videos were taken in both indoor and outdoor environments, which we used to analyze several parameters in our algorithm. During preprocessing, we place depth frames with half the image resolution at the bottom of the original frame and encode a single H.265 video for each sequence. Note that the following analysis takes certain frame pairs from each sequence to show different effects, while the real performance can be better assessed by the accompanying supplementary video and application materials.

Cars The first scene was recorded walking straight on the sidewalk in the inner city for about one minute. Two reference frames are shown in the first row of Figure 9. The scene features multiple cars and multistory buildings, in addition to a plain and far-reaching ground surface. The scene is special in the sense that it also features many reflective materials (*i.e.,* car windows) and both objects, which are very close and very far from the actual view point.

A noticeable difference between the two eyes can be observed in Figure 10a, which is due to the camera offset, but also to the perspective induced by our effect (*i.e.,* the gray border of the sign is visible only in the left eye). Our effect can also nicely reproduce occlusions, as seen from the bicycle, which is more occluded by the vending machine in the view of the right eye.

Barn The second scene, as shown by two representative frames in the second row of Figure 9, shows an old barn in the countryside, which is used to store old wood and machinery. The main characteristic is the poor lighting conditions, in combination with very dark materials in the surroundings (*i.e.,* wood). The duration of the video is a little less than two minutes and also includes a recording of the tractor cabin inside.

In Figure 10b, on the one hand, the strong disparity effect between the two eyes is shown if the foreground and background objects are far apart. On the other hand, on closer inspection, the disruptive repetitive pattern around the edge (*i.e.,* the depth discontinuities mentioned in Section 3.2) is shown in the image of the left eye, while it is well hidden in the image of the right eye.

Forest The third and longest scene was recorded during a hike in the woods for about 3 hours. The third row in Figure 9 shows two frames of this sequence, which feature different lighting conditions, fine-grained textures, and bumpy camera movements. The sequence was shortened to about 1.5 hours to incorporate various conditions encountered during the hike, but still to allow testing on large amounts of data and long video recordings.

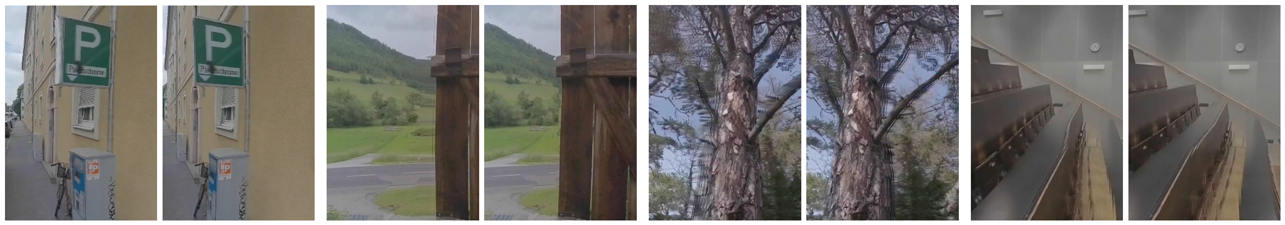
(a) Ticket machine from the cars scene. (b) Barn door from the barn scene. (c) Tree branches from the forest scene. (d) Benches from the lecture scene.

Figure 10: A selection of parts of each scene showing the view of the left and right eye respectively: The disparity between each left and right image is clearly noticeable in every example.

In the frames shown in Figure 10c, a strong disparity effect is shown in a particular view of a tree close to the camera position. In addition, repetitive patterns around the edges of the tree are very clearly visible. These are mainly a result of depth map inaccuracies due to very thin tree branches, which are not represented accurately in the corresponding depth map, which is admittedly provided in a lower resolution. An alternative improvement could be provided by the use of a full-resolution depth map, or the use of a successor to Depth Anything with improved depth estimation capabilities.

Lecture Our final scene was recorded in a lecture hall, as depicted from two exemplary images in the last row of Figure 9. This scene is about two minutes long and very special due to the dominance of bland textures and straight lines and edges. Such structures primarily challenge the capabilities of the underlying depth estimation algorithm, but also our spherical visualization setup.

In Figure 10d, two frames are shown for the left and right eyes, respectively. Although the edge of the table is a straight line, apparently the transition between spheres of multiple distinct radii becomes clearly visible. We attribute these distortions mainly to the discretization of the, otherwise continuous, depth estimates to a limited number of spheres. Increasing the number of spheres and thereby decreasing the discretization error would clearly reduce the visibility of such artifacts.

5.3 Performance

In order to provide an overview of the overall performance of our method, we evaluated both the offline preprocessing stage and the runtime stage separately.

In Figure 11, the time required for the estimation of the depth map is given based on a polling size of 1/16 of the original image resolution, anticipating the slicing (*i.e.,* segmentation) of an individual large video into segments of 30 frames each. Apparently, this segmentation step takes the most time individually due to an increased computation and memory access when saving and creating video clips. However, most of the time is spent on the depth map calculation, as this has to be done for each individual frame.

The individual time to estimate the depth map based on different polling sizes is shown in Figure 12. Obviously, the polling resolution strongly dictates the speed of the depth computation and, at the same time, also indicates the size and accuracy of the overall depth map.

We tested our application on two VR headsets with notably different hardware specifications, the Meta Quest 2 and the Meta Quest 3. We tracked their performance on the basis of a varying number of spheres and depict the results in Figure 13.

The maximum frame rate achieved was 72fps, which is due to a manufacturer limitation[5]. However, every value below 72fps thereby actually suggests full utilization of the hardware. Although the quality of the effect increases with an increase in the number of spheres, we consider 50fps to be the lower limit without significant negative impact, such as headache and nausea, following the findings of Zhang [31]. In turn, this means that the maximum number of spheres on our tested headsets is 8 and 16 for Meta Quest 2 and Meta Quest 3, respectively.

[5]Meta Performance Governor

6 Conclusion and Future Work

In this work, we present our approach to recreate depth experience for 360° monocular panoramic videos in VR. Our approach relies on deep learning, Depth Anything, which infers metric depth to imagery based on contextual clues. In our method, we use this framework to enhance individual monocular 360° images with depth estimates, and we provide an approach to create an immersive replay in VR using a multi-spherical setup. We experimentally investigate several aspects of our approach and provide examples to assess the performance of our method.

Based on the experimental evaluation, several areas for further improvement become apparent. An example of such an improvement is the use of a more evolved depth estimation pipeline, Depth Anything v2 [30], which became available just recently. As this new version claims better performance for thin detailed structures, problems, as mentioned in the hiking scenario, might be significantly reduced automatically. Another obvious improvement involves the number of spheres that can be rendered in a conventional consumer headset. Improving the overall implementations in one way or the other and thereby increasing the number of spheres certainly increases immersion and reduces problems, as mentioned, for example, for the lecture hall scene. Finally, separating depth and RGB video as two separate video streams would allow us to provide full-frame depth imagery, rather than down-scaled ones. This type of adaption of our pipeline is merely of engineering character; however, it is important to understand the implications on the overall process. This concerns the additional overhead and synchronization efforts induced by using multiple video player instances and an increased amount of depth data onto the runtime performance on state-of-the-art headsets. We therefore consider this as work further down the road, while for the time being we focus on the reduction of visual artifacts at depth discontinuities by testing more evolved blending algorithms.

Acknowledgments

Funded by the European Union under Grant Agreement No. 101092861. Views and opinions expressed are, however, those of the author(s) only and do not necessarily reflect those of the European Union or the European Commission. Neither the European Union nor the granting authority can be held responsible for them.

References

[1] D. Anguelov, C. Dulong, D. Filip, C. Frueh, S. Lafon, R. Lyon, A. Ogale, L. Vincent, and J. Weaver. Google Street View: Capturing the World at Street Level. *Computer*, 43(6):32–38, 2010. 2

[2] B. Attal, S. Ling, A. Gokaslan, C. Richardt, and J. Tompkin. MatryODShka: Real-time 6DoF Video View Synthesis Using Multi-sphere Images. In *ECCV*, p. 441–459, 2020. doi: 10.1007/978-3-030-58452-8_26 2

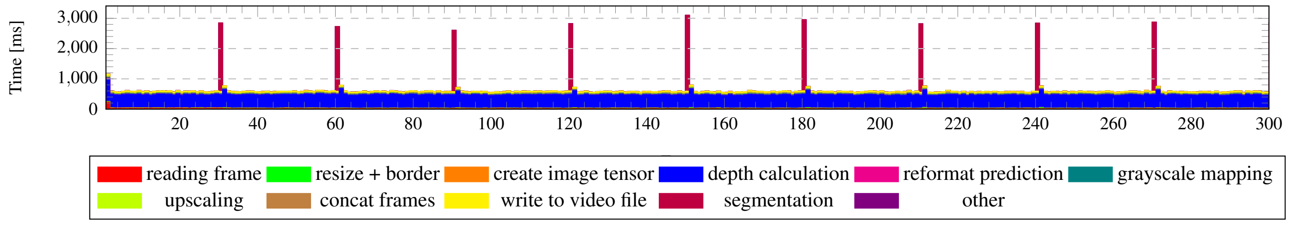

Figure 11: Time analysis of individual parts of the video depth pipeline over the duration of 300 frames: The highest individual time consumption is at the segmentation into short clips. However, this only happens once per 30 frames, so the highest over all time consumption is at the depth calculation step.

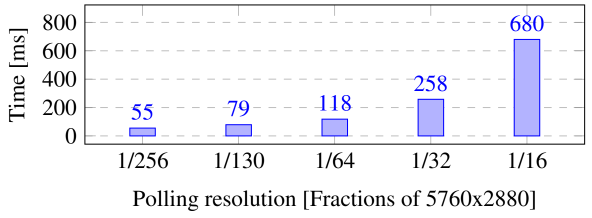

Figure 12: Computation time per frame depending on polling size: The polling size is once given in pixel resolution and once in a fragment of the original image.

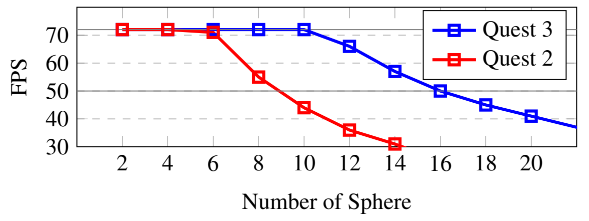

Figure 13: Performance Review (5.7k resolution @ 30FPS): The Meta Quest 3 headset is able to handle almost twice the amount of spheres compared to the Meta Quest 2 headset at the same frame rate.

[3] S. F. Bhat, R. Birkl, D. Wofk, P. Wonka, and M. Müller. Zoedepth: Zero-shot Transfer by Combining Relative and Metric Depth. *arXiv preprint arXiv:2302.12288*, 2023. 2

[4] M. Broxton, J. Flynn, R. Overbeck, D. Erickson, P. Hedman, M. Duvall, J. Dourgarian, J. Busch, M. Whalen, and P. Debevec. Immersive Light Field Video with a Layered Mesh Representation. *TOG*, 39(4):86–1, 2020. 2

[5] C. Coles-Brennan, A. Sulley, and G. Young. Management of Digital Eye Strain. *Clinical and experim. Optometry*, 102(1):18–29, 2019. 4

[6] V. Gkitsas, V. Sterzentsenko, N. Zioulis, G. Albanis, and D. Zarpalas. PanoDR: Spherical Panorama Diminished Reality for Indoor Scenes. In *CVPR Workshops*, pp. 3711–3721, 2021. doi: 10.1109/CVPRW53098.2021.00412 2

[7] R. I. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge Univ. Press, ISBN: 0521540518, 2nd ed., 2004. 2

[8] R. Ishikawa, H. Saito, D. Kalkofen, and S. Mori. Multi-Layer Scene Representation from Composed Focal Stacks. *TVCG*, 29(11):4719–4729, 2023. doi: 10.1109/TVCG.2023.3320248 4

[9] Y. Jiang, C. Yu, T. Xie, X. Li, Y. Feng, H. Wang, M. Li, H. Lau, F. Gao, Y. Yang, and C. Jiang. VR-GS: A Physical Dynamics-Aware Interactive Gaussian Splatting System in Virtual Reality. In *ACM SIGGRAPH*, 2024. doi: 10.1145/3641519.3657448 2

[10] H. Kato, Y. Ushiku, and T. Harada. Neural 3D Mesh Renderer. In *CVPR*, pp. 3907–3916, 2018. 2

[11] B. Ke, A. Obukhov, S. Huang, N. Metzger, R. C. Daudt, and K. Schindler. Repurposing diffusion-based image generators for monocular depth estimation. In *CVPR*, pp. 9492–9502, 2024. 3

[12] M. Kennedy, S. Kopp, et al. *Understanding Map Projections*, vol. 8. Esri Redlands, CA, 2000. 2

[13] G. Kim, D. Kim, J. Jang, and H. Hwang. Pair360: A paired dataset of high-resolution 360° panoramic images and lidar scans. *RA-L*, 9(11):9550–9557, 2024. doi: 10.1109/LRA.2024.3460418 2

[14] D. Lars and P. Anthony. Game engines on Steam: The definitive breakdown. https://www.gamedeveloper.com/business/game-engines-on-steam-the-definitive-breakdown, 2022. Accessed: 2024-03-26. 5

[15] F. Leberl, A. Irschara, T. Pock, P. Meixner, M. Gruber, S. Scholz, and A. Wiechert. Point Clouds: Lidar versus 3D Vision. *Photogrammetric Engineering & Remote Sensing*, 76(10):1123–1134, 2010. 2

[16] Z. Li, S. F. Bhat, and P. Wonka. PatchFusion: An End-to-End Tile-Based Framework for High-Resolution Monocular Metric Depth Estimation. In *CVPR*, pp. 10016–10025, 2024. 3

[17] D. Liu, P. An, R. Ma, W. Zhan, and L. Ai. Scalable Omnidirectional Video Coding for Real-time Virtual Reality Applications. *IEEE Access*, 6:56323–56332, 2018. 2

[18] D. Martins, K. Van Hecke, and G. De Croon. Fusion of Stereo and Still Monocular Depth Estimates in a Self-Supervised Learning Context. In *ICRA*, pp. 849–856, 2018. doi: 10.1109/ICRA.2018.8461116 2

[19] K. Matzen, M. F. Cohen, B. Evans, J. Kopf, and R. Szeliski. Low-cost 360 Stereo Photography a. Video Capture. *TOG*, 36(4):1–12, 2017. 2

[20] M. Mühlhausen, M. Kappel, M. Kassubeck, L. Wöhler, S. Grogorick, S. Castillo, M. Eisemann, and M. Magnor. Immersive Free-Viewpoint Panorama Rendering from Omnidirectional Stereo Video. *Computer Graphics Forum*, 42(6):e14796 ff., 2023. doi: 10.1111/cgf.14796 2

[21] R. Nevatia. Depth Measurement by Motion Stereo. *Computer Graphics and Image Processing*, 5(2):203–214, 1976. 2

[22] R. Ranftl, K. Lasinger, D. Hafner, K. Schindler, and V. Koltun. Towards Robust Monocular Depth Estimation: Mixing Datasets for Zero-shot Cross-dataset Transfer. *PAMI*, 44(3):1623–1637, 2020. 2

[23] H. Shum and S. B. Kang. Review of Image-based Rendering Techniques. In *Visual Communications and Image Processing*, vol. 4067, pp. 2 – 13, 2000. doi: 10.1117/12.386541 2

[24] S. Ueda, H. Saito, and S. Mori. Toward Multi-Plane Image Reconstruction from a Casually Captured Focal Stack. In *VISAPP*, pp. 72–82. INSTICC, SciTePress, 2024. doi: 10.5220/0012438700003660 2

[25] J. Unger, A. Wenger, T. Hawkins, A. Gardner, and P. Debevec. Capturing and Rendering with Incident Light Fields. In *Eurographics*, p. 141–149, 2003. 3

[26] Q. Wang, Z. Wang, K. Genova, P. Srinivasan, H. Zhou, J. T. Barron, R. Martin-Brualla, N. Snavely, and T. Funkhouser. IBRNet: Learning Multi-View Image-Based Rendering . In *CVPR*, pp. 4688–4697, June 2021. doi: 10.1109/CVPR46437.2021.00466 4

[27] Y. Wang, P. Wang, Z. Yang, C. Luo, Y. Yang, and W. Xu. UnOS: Unified Unsupervised Optical-flow and Stereo-depth Estimation by Watching Videos. In *CVPR*, pp. 8071–8081, 2019. 2

[28] Z. Wang. 3D Representation Methods: A Survey. *arXiv preprint arXiv:2410.06475*, 2024. 2

[29] L. Yang, B. Kang, Z. Huang, X. Xu, J. Feng, and H. Zhao. Depth Anything: Unleashing the Power of Large-scale Unlabeled Data. In *CVPR*, pp. 10371–10381, 2024. 1, 2, 3

[30] L. Yang, B. Kang, Z. Huang, Z. Zhao, X. Xu, J. Feng, and H. Zhao. Depth Anything V2. *arXiv preprint arXiv:2406.09414*, 2024. 7

[31] C. Zhang. Investigation on Motion Sickness in Virtual Reality Environment from the Perspective of User Experience. In *ICISCAE*, pp. 393–396. IEEE, 2020. 7