Exemplar-Based Inpainting for 6DOF Virtual Reality Photos

Shohei Mori, Dieter Schmalstieg, and Denis Kalkofen

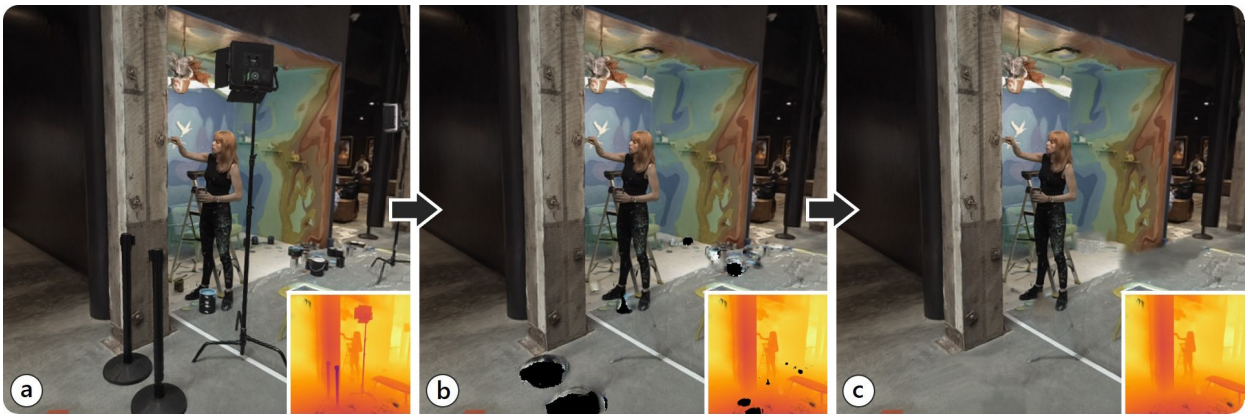

Fig. 1: Multi-layer image inpainting for immersive VR photos (depth rendering in insets), in the Alexa Meade Exhibit scene [7]. (a) Multi-layer scene representations enable immersive VR in real time that supports wide view, stereoscopy, and full 6DOF head motion. (b) We first interactively delete objects by masking and then (c) fill in remaining holes with an inpainting algorithm designed for such layered scene data structure.

Abstract—Multi-layer images are currently the most prominent scene representation for viewing natural scenes under full-motion parallax in virtual reality. Layers ordered in diopter space contain color and transparency so that a complete image is formed when the layers are composited in a view-dependent manner. Once baked, the same limitations apply to multi-layer images as to conventional single-layer photography, making it challenging to remove obstructive objects or otherwise edit the content. Object removal before baking can benefit from filling disoccluded layers with pixels from background layers. However, if no such background pixels have been observed, an inpainting algorithm must fill the empty spots with fitting synthetic content. We present and study a multi-layer inpainting approach that addresses this problem in two stages: First, a volumetric area of interest specified by the user is classified with respect to whether the background pixels have been observed or not. Second, the unobserved pixels are filled with multi-layer inpainting. We report on experiments using multiple variants of multi-layer inpainting and compare our solution to conventional inpainting methods that consider each layer individually.

Index Terms—Multi-layer images, inpainting, virtual reality, image-based rendering

1 Introduction

Viewing natural scenes in virtual reality (VR) requires not only a wide viewing range, but also support for six degrees of freedom (6DOF) head motion to make the experience comfortable [62]. Currently, the most prominent scene representation fitting these requirements is the family of multi-layer image formats [2, 7, 36]. In a multi-layer image, each layer consists of pixels with color and transparency channels; pixels are most opaque in the layer containing the most likely depth of the scene content. Each layer partly occludes the subsequent layers or represents a partially translucent material [61]. Therefore, stacking the layers results in a full scene representation with plausible disparities. The layers are reconstructed from a depth volume analysis using multi-view depth maps [49] or, more recently, neural networks [16, 38, 70].

Like with conventional photography, the need for image editing arises in image-based content for VR viewing. We are especially interested in the removal of obstructive objects, such as cars, trash bins in a park, or one's ex-spouse. Conventional single layer image editing on the composited image before presentation could be used, but loss of all depth information at this point in the imaging pipeline would make achieving correct stereoscopy and temporal coherence exceedingly difficult [60]. Editing the individual layers before compositing removes unwanted pixels by making them fully transparent, so that compositing fills in view-dependent background pixels.

However, areas unobserved in all of the original multi-view images can only fall back to a default background color, leading to unpleasant "black holes." For these unobserved areas, we desire to automatically fill in plausible pixel colors and transparencies. Previous approaches attempt to inpaint the original multi-view images; such multi-view inpainting has been formulated only for plane proxies [50], light field data in a regular grid [31], structure from motion data [3, 63], and runtime scene reconstruction [41, 42]. To the best of our knowledge, we are the first to tackle the problem of multi-layer image inpainting of missing content from unobserved areas.

We address this issue with a two-step inpainting system. Our system allows the user to interactively crop multi-layer regions and supports determining regions to be inpainted by occlusion calculation. We formulate the multi-layer image inpainting problem as an approximated nearest neighbor field (ANNF) search in image space, which defines patch locations used to fill in holes, followed by blending of the gathered patches, in analogy to conventional image inpainting. In summary, we present the following contributions to multi-layer image inpainting:

- We present the first approach to inpainting multi-layer scene representations for hole filling.
- We introduce a multi-layer image inpainting pipeline that includes an interactive segmentation to expose background pixels and a

• *Shohei Mori, Dieter Schmalstieg, and Denis Kalkofen are with Graz University of technology. E-mail: s.mori.jp@ieee.org.*

Manuscript received xx xxx. 201x; accepted xx xxx. 201x. Date of Publication xx xxx. 201x; date of current version xx xxx. 201x. For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org. Digital Object Identifier: xx.xxxx/TVCG.201x.xxxxxxx

multi-layer image inpainting algorithm that fills in unobserved remaining pixels.

- We compare variants of our multi-layer image inpainting algorithm with baseline approaches to validate our design choices.

2 Related work

This section gives an overview of VR photography and of inpainting. It provides the motivation for choosing multi-layer images over other possible scene representations for inpainting. We also discuss possible extensions of existing approaches to emerging multi-layer data structures and compare the resulting problems to the ones encountered in conventional inpainting.

2.1 Photography for VR

Possibly the most successful VR application to date is immersive presentation of photographic content. Oftentimes, this content is based on light fields [18, 33] to support stereoscopic viewing, wide field of view, high-speed rendering, and unrestricted 6DOF head motion [7, 44].

The core idea of light field rendering is to organize multi-view inputs in a structure with more than $n = 2$ dimensions and blend them along light rays passing through the lens of a simulated camera [20, 64]. Therefore, the dimensionality of the data structure determines the degrees of freedom of head motion in scene content viewing. Structured $n \leq 3$ data (e.g., omnidirectional stereo [21] and concentric mosaics [57]) support only stereoscopic views under restricted panning or 2D locomotion, while light fields with $n = 4$ enable 6DOF head motion [33], albeit at a high memory cost.

Since the rendering process of such data includes multi-ray or multi-view blending, the results tend to be blurry except at focused surfaces [40]. Such synthetic aperture artifacts are well-accepted in the context of lens camera simulation [20, 64]. However, in commodity VR displays, the user's eyes are often modeled as a pinhole cameras, and such cameras should not have any defocus blur[1]. Additional depth information helps to focus the rays back to the known depth [8, 14, 44] or enables the direct projection of pixels to the depth points [55]. However, storing a discrete scene depth per pixel does not make sense for transparent surfaces, which are well represented in the original light field rendering approach without depth information [32]. Further, online view selection [8] and visibility evaluation [14] can be costly for VR displays operating at high spatial resolutions and temporal refresh rates. The use of per-view geometry proxies partially works around this problem [13, 44], although overlapping proxies for smooth view interpolation show double images at wrong depths.

Recent work in VR photography [7, 36, 38] instead uses layered geometry proxies uniformly arranged in diopter space [61] and combines them with per-layer alpha buffers to encode the depth uncertainty of the observed scene points [49]. This kind of representation lends itself to analysis by a neural network within a plane-sweep or sphere-sweep volume formed by multiple calibrated images [2, 16, 70] or a focal stack [22]. The resulting multi-layer images support 6DOF head motion and can be rendered efficiently via back-to-front compositing using a conventional "over" blending operator. Ambiguous scene points become redundant pixels at different layers, thereby preventing unintended revealing of scene portions occluded behind objects. Depending on the layers' geometric shape, multi-layer images are either known as multi-plane image (MPI) or multi-sphere image (MSI).

Unlike an MPI, a single MSI [7] can cover the entire field of view, thereby avoiding the problems caused by the need to handle groups of multi-view images consistently. Specifically, a region of interest (ROI) needs to be labeled only once (although semi-automated structured light field segmentation [25] could possibly be used for this purpose). Moreover, no multi-view blending is required as in the MPI case [38], and, therefore, no multi-view inpainting [3, 41, 42] based on unreliable depth information is needed.

[1] Note that we discuss only VR displays with a fixed display focus, but future VR displays may find synthetic aperture rendering in immersive VR photos beneficial. For example, foveated rendering [19, 47] or internal rendering in varifocal displays [15, 48] could benefit as well.

Neural scene representations based on multi-view images [37, 39] or light fields [28, 58] have recently emerged as competitors to traditional image-based models. Especially neural radiance fields (NeRF) [39] have received a lot of attention. However, the significant scene-specific training time as well as the implicit scene representation of NeRF methods make them rather unsuitable for our use case of interactive scene editing and sampling, although faster turn-around times [43] and a family with voxel representations [1] will be possible in the near future [23].

2.2 Image inpainting

Exemplar-based inpainting algorithms find a collection of similar patches within a given image to fill in specified image holes. Among traditional exemplar-based methods for image inpainting, PatchMatch [4, 5] is probably the most successful, due to its performance and flexibility in catering to various image editing problems (e.g., image retargeting or reshuffling). It is the foundation for the widely used "content-aware" fill in Adobe Photoshop. Its major difference from other inpainting approaches, such as diffusion [52] or region [11] methods, is its efficient ANNF search. Therefore, we use it as a starting point for our multi-layer method.

Inpainting algorithms based on a convolutional neural network (CNN) solve the same problem by learning a masked context rather than collecting pixels to fill holes elsewhere in the image. Reliance on context rather than exemplars is an advantage if no suitable exemplars are present in the target image, but a clear disadvantage if insufficient training examples (i.e., similar images) are available. Therefore, some variants use the exemplar idea in the sense of contextual attention [67] and feature-space patch swapping [59] to preserve high-frequency and coherent content.

Both inpainting methods based on exemplars [9] and on a CNN [46] have demonstrated their ability to fill larger holes. Both methods work well with natural scenes, such as dirt, grass, trees, etc., but have difficulties with purpose-made environments.

These existing methods could be used for multi-layer image inpainting by inpainting objects in the original multi-view images and then generating multi-layer images from the intermediary results. For this purpose, one needs a multi-view labeling tool and an inpainting algorithm that can produce consistent inpainting over multiple images of the same scene [3, 41, 42] despite incomplete or unstable depth information connecting the pixels in these images. Alternatively, a new algorithm may be designed to work directly on a multi-layer image with color and transparency channels. In our experiments, we compare these two alternatives.

2.3 3D inpainting

Because of its implicit depth information, a multi-layer image can be rendered from an arbitrary viewpoint, like a conventional 3D model (e.g., a textured mesh or an irregularly sampled volume). Unfortunately, inpainting a 3D model is more challenging than inpainting a 2D image. Some specialized methods exist. Textured mesh completion may fill a hole with a similar mesh [45]. Volumetric inpainting is used to complete geometric scans [12, 34] or computational tomography [35].

Unlike explicit 3D models, multi-layer images are restricted to discrete layers [49]. Consequently, they do not require any geometry completion. However, the layers are arranged at discrete distances from a center of projection, making image resolution dependent on the layer. Consequently, volumetric inpainting methods designed for a uniform voxel grid or a light field space [31] do not fit multi-layer image inpainting.

The most closely related to our work in terms of the underlying data structure are methods that partially fill in the background layers [24, 54, 56, 60]. However, the use case pursued by these methods is noticeably different from ours. Since these algorithms are designed to only extend the boundaries, attempting to remove an object from the middle of the scene will create difficulties if empty pixels must be filled. Besides, some methods rely on depth information [24, 56], which tends to cause distortion in multi-view applications. Therefore, these approaches are less effective than our multi-layer image representation

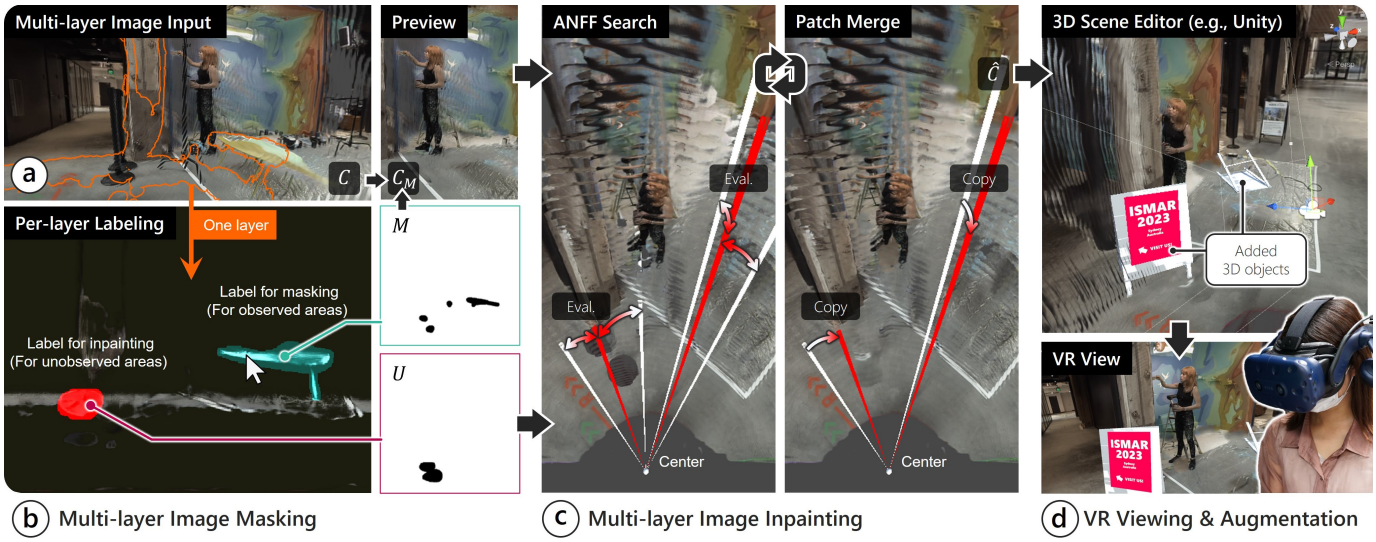

Fig. 2: Pipeline of our multi-layer image inpainting. (a) Given a complete multi-layer image, (b) we segment out objects by newly assigning alpha values to the pixels. The intermediate multi-layer image is rendered in a preview window, so that we can confirm the current inpainting results in 3D. As front objects are removed, background pixels may pass through the multi-layer image completely. Such unobserved pixels are automatically selected and filled in at the next inpainting stage. (c) Given the masked multi-layer image, our inpainting algorithm fills in the remaining areas to complete the multi-layer image. (d) Finally, one may further edit the completed multi-layer image by placing 3D objects.

(see Section 2.1). The only work that uses multi-layer images [60] relies on visible front pixels, which are, in our case, empty and cannot be anticipated before identifying the inpainted structures[2]. In multi-layer images, a large portion of the background tends to be well observed and encoded in other layers. We only need to apply inpainting to the unobserved areas which cannot be filled in from farther layers. These unobserved areas will usually be small, but need to be handled in a manner that is consistent across all layers of the multi-layer image.

3 Method

In this section, we give an overview of our multi-layer image inpainting pipeline, including data structures and data flow. The pipeline consists of the following four stages, as illustrated in Fig. 2:

(a) Acquisition of input multi-layer image (MPI or MSI)

(b) Labeling of the ROI

(c) Inpainting of the ROI

(d) VR viewing of the inpainted result

Multi-layer image input. Our input data is a multi-layer image (Fig. 2a) with a spatial resolution of $N_x \times N_y$ and N_z layers, where the farthest layer has index 1. Each pixel has a color value $C(\mathbf{p}) \in \mathbb{R}^3$ and a transparency $\alpha(\mathbf{p}) \in \mathbb{R}$ at position $\mathbf{p} = [b_x, b_y, b_z]^\top \in \mathbb{P}$, where $\mathbb{P} \equiv [1..N_x] \times [1..N_y] \times [1..N_z]$. One may generate such a multi-layer image directly from calibrated multi-view images [7], omni-directional stereo images [2], or by converting an MPI representation into MSI.

Multi-layer image labeling. We provide an interactive painting tool that lets the user rapidly segment a volumetric ROI by sweeping through the layers and painting the ROI into a per-layer mask M in each 2D layer (Fig. 2b). M is a multi-layer alpha mask image indicating the ROI with continuous values in the range $[0,1]$, which is set to 1 upon the application initialization and specified to 0 for complete masking. Our tool uses a split screen: A first window shows the layer where the user is currently painting the mask M, and a second window shows the resulting multi-layer scene reconstruction. Thus, the second window provides a preview of the ROI removal without inpainting, i.e., $C_M = C \cdot M$ and $\alpha_M = \alpha \cdot M$ for color and alpha channels, respectively (Fig. 1b). Fig. 3 shows an example case of masking the black poles in Fig. 1a using our tool.

[2]Please refer to the supplemental video for examples

A second binary mask U can be computed from α_M for those areas where inpainting needs to fill in synthetic content. For this purpose, we define the set of all possible view rays $\mathbf{r} \in R(\mathbf{p})$ which pass through a sample point $\mathbf{p}$. As a ray $\mathbf{r}$ traverses the multi-layer image, it accumulates opacity (aka over-composed alpha [51]). Given an opacity threshold τ, let the function $s_\tau(\mathbf{r})$ compute the index of the layer where the threshold is reached, or zero, if the ray leaves the volume covered by the multi-layer image without reaching the threshold. The latter case describes an *unobserved ray*, which requires inpainting at least one of the samples it draws from. A sample is marked in U if it is pierced by at least one unobserved ray (which makes it a candidate for inpainting), but it does not contribute to any observed ray (changing the sample does not affect other rays).

$$U(\mathbf{p}) = \begin{cases} 0, & \text{if } \exists \mathbf{r} \in R(\mathbf{p}) \text{ s.t. } s_\tau(\mathbf{r}) = 0 \\ & \wedge\ \nexists \mathbf{r} \in R(\mathbf{p}) \text{ s.t. } s_\tau(\mathbf{r}) \geq b_z \\ 1, & \text{otherwise.} \end{cases} \tag{1}$$

Multi-layer image inpainting. Given C_M, α_M and U, our inpainting algorithm fills the specified areas in the multi-layer image and outputs an inpainted multi-layer image, $\hat{C}$ (Fig. 2c). Our inpainting algorithm is based on an ANNF calculation [4,5] using multiple patch error metrics [26,27]. We explore how different combinations of metrics contribute to the quality of multi-layer image inpainting.

Scene viewing and editing. Finally, we render the output multi-layer image, $\hat{C}$, from back to front with "over" alpha compositing [38], followed by pixel normalization with the projected alpha values [7]. In our VR application, the user may place 3D objects in the rendered scene to further alter the scene. In Fig. 2d, we placed two signboards at the locations where the two black poles and cans were originally located.

3.1 Approximated nearest neighbor field search

Given the masked color layers and mask layers, we fill in the specified ROI to complete the multi-layer image without the visual obstacles, $\hat{C}$. The goal here is to find a bidirectional ANNF, i.e., a mapping f that describes similar patches in the source and target image.

Since our multi-layer images are uniformly spaced in diopters, the spatial resolution between samples in the same layer becomes larger for farther away layers. Hence, copying pixels to a different layer

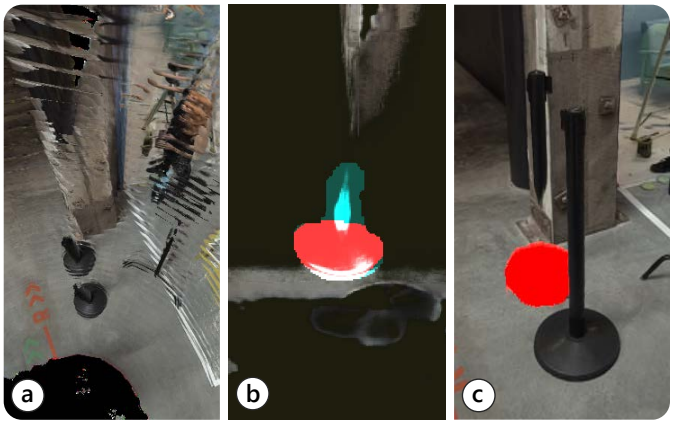

Fig. 3: Labeling in a multi-layer image. (a) We remove the black poles whose background areas are encoded into the rear layers. (b) In a layer image, we consider two types of labels: One for removing frontal objects (green) and another for specifying areas to be inpainted by our multi-layer image inpainting algorithm (red). The latter areas can be automatically calculated, as demonstrated here. (c) The specified area (red) is encoded in the multi-layer mask and can be previewed instantly.

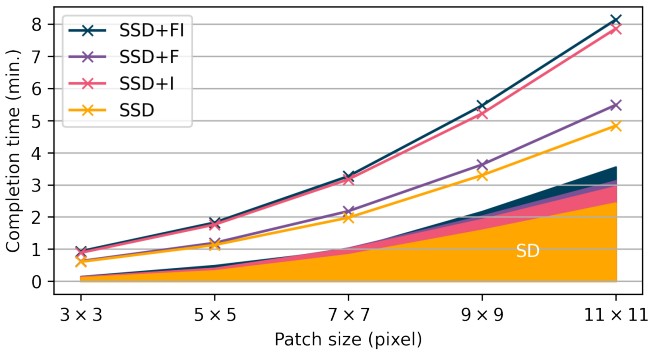

Fig. 4: Mean processing time over the scenes in our dataset. We evaluated the performance of our multi-layer image inpainting algorithm with different patch metrics and patch sizes. Adding the intensity compensation requires more time than adding the patch flipping. The processing time increases with increased patch sizes and varies depending on the size of mask areas as shown in the standard deviations (SD).

requires resampling. To minimize quality loss due to resampling, we only allow pixels to be copied to the same layers. Specifically, we do not copy patches to different depths, but rather copy a volumetric patch. Therefore, we hereafter use $\mathbf{p}$ and $\mathbb{P}$ for 2D locations without loss of generality. An element of our ANNF, $f(\mathbf{p})$, consists of a 2D reference location $f_{\mathrm{L}}(\mathbf{p})$ from where the patch is copied, and a flipping matrix $f_{\mathrm{F}}(\mathbf{p}) \in diag(\pm 1, \pm 1)$ [27] to increase the available candidates to choose from. $f_{\mathrm{F}}(\mathbf{p})$ represents non-flipping, horizontal flipping, vertical flipping, and point reflection. Since a straight line in an MSI is sliced in a spherical volume, such a line only partially appears in flipped locations in each layered image. Therefore, we expect that the flipping by f_{F} can preserve the shape of a line when extended by copying the structure, which could lead to an increase in the number of patch candidates. We find an optimized mapping f^* based on the PatchMatch algorithm [4,5] as

$$f^* = \arg\min_f \sum_{\mathbf{p}\in\mathbb{P}} \mathcal{R}(\rho(f,\mathbf{p})), \tag{2}$$

where $\mathcal{R}(\cdot)$ is a robust loss function (we use Tukey's bisquare loss function [6]), and ρ is a texture dissimilarity term which minimizes the appearance difference between a patch at $\mathbf{p}$ in the target image and a patch referenced at $f(\mathbf{p})$ in the masked image (i.e., C_M). The texture dissimilarity is a weighted sum of a color dissimilarity term $\rho_C(f,\mathbf{p})$ and a transparency dissimilarity term $\rho_\alpha(f,\mathbf{p})$:

$$\rho(f,\mathbf{p}) = w\cdot\rho_C(f,\mathbf{p}) + (1-w)\rho_\alpha(f,\mathbf{p}) \tag{3}$$

The color dissimilarity is evaluated over a symmetric $V\times V$ pixel patch $\mathcal{V} = \{[x,y]^\top \mid -V/2 \le \{x,y\} \le V/2\}$, i.e.,

$$\rho_C(f,\mathbf{p}) = \sum_{\mathbf{v}\in\mathcal{V}} ||C_M(\mathbf{p}+\mathbf{v}) - B(\mathbf{p})C_M(f_{\mathrm{L}}(\mathbf{p}) + f_{\mathrm{F}}(\mathbf{p})\cdot\mathbf{v})|| \tag{4}$$

It has been demonstrated [26] that

$$B(\mathbf{p}) = \frac{\sum_{\mathbf{v}\in\mathcal{V}} C_M(\mathbf{p}+\mathbf{v})}{\sum_{\mathbf{v}\in\mathcal{V}} C_M(f_{\mathrm{L}}(\mathbf{p}) + f_{\mathrm{F}}(\mathbf{p})\cdot\mathbf{v})} \tag{5}$$

cancels out the intensity differences between pairs of patches. In multi-layer image inpainting, we expect that such intensity compensation can also increase the number of patch candidates, since patches can be sampled over the wide field of view. Finally, the transparency dissimilarity is evaluated over the same patch:

$$\rho_\alpha(f,\mathbf{p}) = \sum_{\mathbf{v}\in\mathcal{V}} ||\alpha_M(\mathbf{p}+\mathbf{v}) - \alpha_M(f_{\mathrm{L}}(\mathbf{p}) + f_{\mathrm{F}}(\mathbf{p})\cdot\mathbf{v})||. \tag{6}$$

Note that the dissimilarity calculations (eqs. 4–6) are performed in every layer. We explore the impact of the different patch metrics – naïve sum of squared differences (SSD), flipping estimation, intensity compensation, and their combination – in the evaluations in Section 4.

3.2 Merging multi-layer patches

Calculating f also generates a cost volume ρ that has the same dimensions as the input. We use ρ to gather weighted patches at each layer:

$$\hat{C}(f,\mathbf{p}) = \frac{\sum_{\mathbf{v}\in\mathcal{V}} L(\rho(\mathbf{p}+\mathbf{v}))B(\mathbf{p}+\mathbf{v})C(f_{\mathrm{L}}(\mathbf{p}+\mathbf{v}) - \mathbf{v}\cdot f_{\mathrm{F}}(\mathbf{p}+\mathbf{v}))}{\sum_{\mathbf{v}\in\mathcal{V}} L(\rho(\mathbf{p}+\mathbf{v}))B(\mathbf{p}+\mathbf{v})}, \tag{7}$$

where $L(\cdot)$ is a logarithmic function to convert a SSD cost to a score.

Updating uses an image pyramid consisting of C_M, α_M, U, and f components. Since our goal is to find the best ANNF f in multi-layer image space, we raster-scan the 2D space to update ANNF at each pyramid level as in conventional 2D image inpainting algorithms (i.e., in each iteration, from the top left corner to the bottom right corner and vice versa).

For initialization, we randomly sample locations and flipping matrices over the pixel locations of $U = 0$ at the highest pyramid level. During the raster-scans, we check for better patch candidates at adjacent pixel locations (propagation) and at random sample locations. If such a better candidate is found, the current 2D reference location and the flipping matrix are updated to refer to the candidate. At the end of each iteration, we update $\hat{C}$ as in Equation 7.

To calculate a bidirectional ANNF, we repeat the above ANNF searching and patch merging for $S \to T$ and $T \to S$ in each raster-scan. This raster-scan continues at each pyramid level until it reaches a predetermined maximum number of iterations, or the difference of summed costs to the previous iteration exceeds a given threshold. However, we update the ANNF only when we find a lower cost over the depth volume $\sum_{i=1}^{N_z} \rho(f,i,\mathbf{p}))$, which replaces the conventional 2D cost map given in Equation 2. Upon the termination of the iterations, we upsample a higher-level result to the lower level.

4 Results and Evaluation

In this section, we describe the results achieved with our approach and compare them to the state of the art. It is difficult to compare inpainting methods quantitatively, since there is no ground truth for inpainted areas [41, 46]. Even if we remove a known portion of the background, we can only expect that inpainting fills the removed area with plausible patterns, but not that the inpainting result will match the original background exactly. Hence, a pixel-wise comparison is

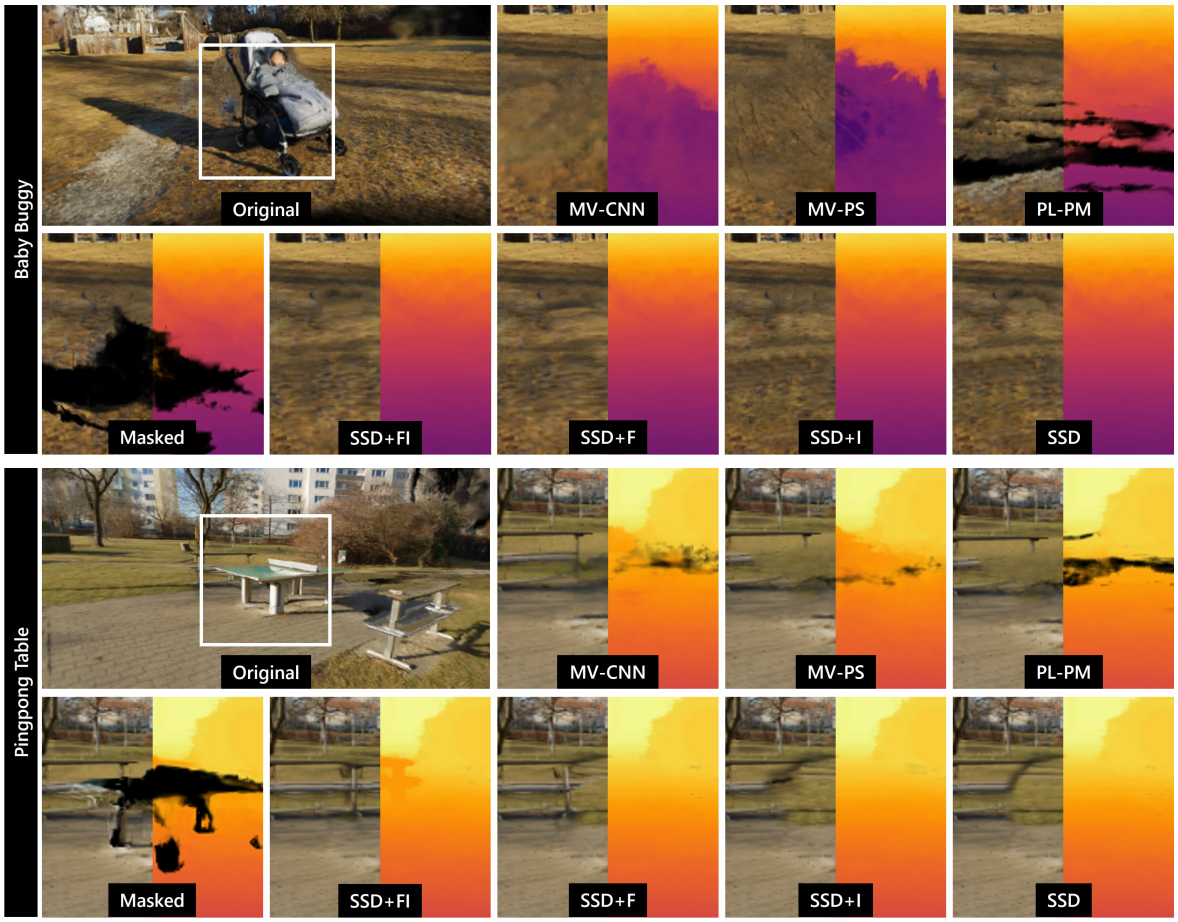

Fig. 5: Qualitative comparison in our original MSI dataset (color and depth domains in left and right halves, respectively). We remove the baby buggy and the ping-pong table from the scenes. The first row shows a rendered frame of an original MSI (Original), two multi-view inpainting results using a CNN [68] and Photoshop content-aware fill (i.e., MV-CNN and MV-PS, respectively), and a per-layer inpainting result using RGB+α PatchMatch [4,5] (PL-PM). The second row shows a rendered frame of a masked MSI (Masked), and four variants of our multi-layer inpainting algorithm (SSD+FI: patch flipping + patch intensity compensation, SSD+F: only patch flipping, SSD+I: only patch intensity compensation, and SSD: naïve SSD). While the three baseline approaches failed to provide inpainting results with consistent colors and depths, our SSD methods successfully fill the inpainting areas. We also provide quantitative results of a user study in Section 4.4.

not a meaningful evaluation method. For this reason, we present the following results: a description of implementation and performance (Section 4.1), a small ablation study testing different configurations of our method (Section 4.2), a visual comparison of our method to the baseline methods (Section 4.3), and a study in which human participants subjectively rated our methods against the baseline methods (Section 4.4).

4.1 Experimental platform

Datasets. We assembled a set of scenes by combining our own scenes and with an existing benchmark dataset. All scenes are composed of real images, representative of real-world use cases of light field photography. We did not consider synthetic scenes, since plausible multi-layer images are difficult to create synthetically – there is no obvious way to convert surface-based computer graphics models into a soft representation of translucent images [49].

For our own scenes, we collected unstructured multi-view photos at seven different outdoor locations. We first calibrate the multi-view photos of each scene using COLMAP [53]. At each viewpoint, we generated an MPI [38] and projected all MPI pixels into the closest MSI layer [36]. Due to the extensive demand for GPU memory and computation, each MPI was restricted to $600 \times 400 \times 96$ pixel resolution. The composed MSI contains $1024 \times 1024 \times 96$ pixels. In addition to our original MSI dataset, we used Google's MSI dataset [7] in the *lowRes* version, which provides image data at $800 \times 600 \times 64$ or $800 \times 600 \times 96$ pixels resolutions and a 3D mesh geometry proxy. The mean and standard deviation values of the numbers of deleted pixels and inpainted pixels over the numbers of all image pixels were $8.71 \times 10^{-3} (2.47 \times 10^{-2})$ and $2.92 \times 10^{-3} (9.61 \times 10^{-3})$, respectively.

Implementation and performance. We implemented the algorithm described in Section 3.1 in C++ and ran it on a desktop computer (AMD Ryzen 7 3700X, 8 cores, 3.59 GHz base clock, 16 GB RAM, Windows 10) connected to a Meta Quest 2 headset via an Oculus link cable for VR viewing. In our implementation, we parallelized independent per-layer operations and ran bidirectional ANNF search using OpenMP. Fig. 4 shows the performance within the dataset. Adding patch flipping and intensity compensation increases the processing time. Especially the intensity compensation requires performing convolutions over the multi-layer images and therefore takes more time than the flipping. While our implementation runs on a multi-core CPU, it could be implemented on a GPU with a highly efficient jump flood algorithm [69] for a further speed up. We leave this optimization for future work.

4.2 Comparison of our multi-layer patch metrics

We compared multiple variants of our multi-layer image inpainting algorithm to investigate the impact of different metrics. All variants used $w = 0.25$ and $V = 5$. The full method, *SSD+FI*, used both the flipping f_F and the intensity correction B. *SSD+I* omitted flipping (i.e., $f_F = I$); *SSD+F* omitted intensity correction (i.e., $B = 1$). A naïve

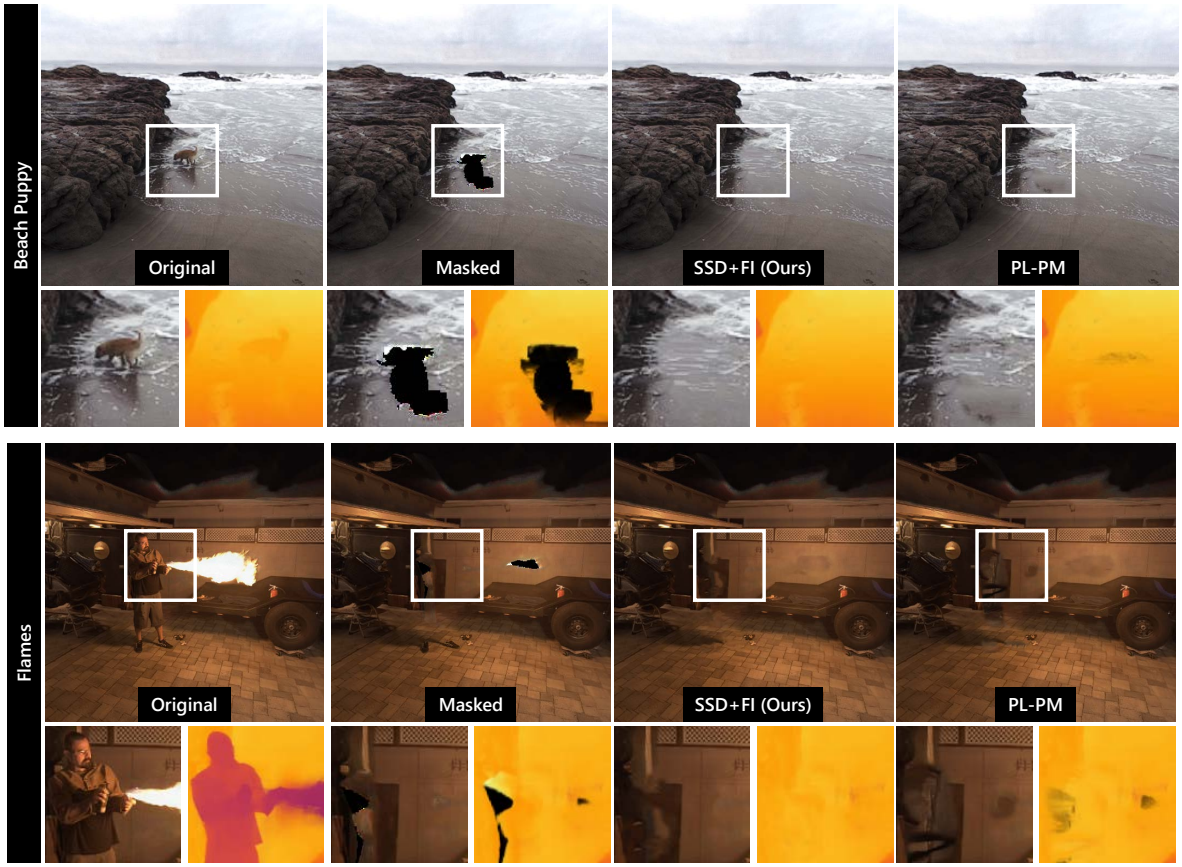

Fig. 6: Qualitative comparison in Google's MSI dataset [7] (color and depth domains side-by-side). (top) We removed the puppy together with its shadow and reflections on the waves. (bottom) We removed the person and the flame. Both scenes have been successfully inpainted by our multi-layer image inpainting algorithm (*SSD+FI*). *PL-PM* presents reasonable inpainting result in the color domain. However, the depth-domain result reveals its incomplete thin layers.

Table 1: Characteristics of the inpainting methods in our comparison

Method	Description	Intensity	Flip	Layers
SSD+FI	Full	B	yes	N_z
SSD+F	No intensity	1	yes	N_z
SSD+I	No flipping	B	no	N_z
SSD	Naïve	1	no	N_z
PL-PM	Single layer	1	no	1
MV-PS	Photoshop	-	-	-
MV-CNN	DeepFill v2	-	-	-

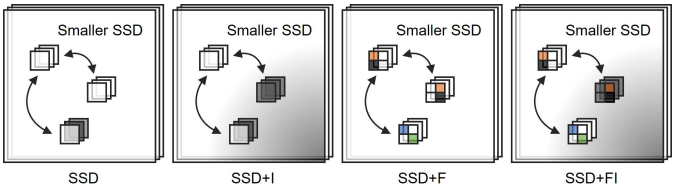

Fig. 7: Visualization of our multi-layer patch metrics to be studied.

method *SSD* used neither flipping nor intensity correction. See Table 1 for a summary of the implementations. Fig. 7 shows a visual summary of the implementations.

Fig. 5 presents qualitative comparisons of the original, masked, and inpainted MSI rendering results in two of the seven scenes in our dataset. Insets are enlargements of the rectangular area indicated in white in the original MSI. Each subset contains inpainting results in the color and depth domains on the left and right half, respectively. We selected these scenes as representative inpainting results. In the Baby Buggy scene, we expected to remove the buggy and recover the natural grass patterns. Similarly, in the Ping-Pong table scene, we wanted to retrieve the metal bench structure after removing the table that partially occludes the bench.

As can be seen in the second row of each scene, the patch metrics variants exhibit apparent differences in their inpainting results. As expected, masking reveals valid portions background layers, especially noticeable in the Baby Buggy scene. In the Baby Buggy scene, *SSD* and *SSD+I* tend to repeat a particular pattern in the inpainted areas due to the lack of patch samples. We observe that repeated patterns are enhanced or less blurred in *SSD+I*, as it can provide intensity-compensated versions of the chosen patches. Artificial patterns become less pronounced in *SSD+F*, which relies on flipping for supplying a wider variety of patch variations. However, the inpainted areas appear stretched horizontally and do not match very well to the surrounding areas. With the *SSD+FI*, these artifacts become less visible. In the Ping-Pong table scene, the flipping function enables the complete recovery of the occluded metallic bench behind the masked table. Subjectively, *SSD+FI* generates the visually most pleasing results. Therefore, we elected *SSD+FI* as our primary method for further comparisons.

4.3 Comparison of our method to sequential methods

Our main contribution is the extension of exemplar-based inpainting to multiple layers in a consistent manner. In the sense of an ablation study, we wanted to investigate how our approach competes with methods that operate sequentially, one image or layer at a time. To this aim, we implemented two sequential methods:

1. Per-layer inpainting. This condition, *PL-PM*, restricts the inpainting algorithm to operate on only one layer at a time. Even without considering coherence across layers, the inpainting method must still be able to handle the alpha buffer of the MSI in addition to the color buffer. Therefore, we implemented this method by parameterizing our exemplar-based method with $N_z = 1, B = 1$ and $f_F = I$.

2. Multi-view inpainting before MSI generation. Unlike the per-layer inpainting, inpainting of the original views before conversion into MSI can work on a single masked color image at a time. This simplified requirement enables us to employ any state-of-the art inpainting algorithm. Hence, we apply two successful algorithms for this pipeline: the content-aware fill in Adobe Photoshop 2023 as the commercial state of the art (*MV-PS*) and DeepFill v2 [68] as a representative CNN baseline (*MV-CNN*). Objects were manually labeled in each input view. While either of these methods is likely to outperform our method in terms of quality achieved on a single image, we expected that per-view inpainting would suffer from inconsistencies across multiple image views and, therefore, wrongly estimated pixel colors and depths in the resulting MSI. As reported in the literature [66], we observed strong artifacts in inpainted results at 600×400 pixels when using DeepFill v2. For those images, we applied 3×3 Gaussian blur before applying the inpainting algorithm, which removed the artifacts.

We considered the method of Srinivasan et al. [60] as a third competitor, but it would require manually masking front-facing pixels required for initialization, which is infeasible to do before the depth structure of the MSI is known. Hence, adopting this method for MSI inpainting is a complex endeavor left for future work.

Comparison on our dataset Refer again to Fig. 5: Both multi-view approaches, *MV-CNN* and *MV-PS*, successfully fill in the area without revealing fallback pixels in the Baby Buggy scene, but fail in the Ping-Pong table scene. The inpainted multi-view images do not contain any fallback pixels, and the MPI generator [38] tries to explain the multi-view images as a complete MPI. However, as we expected, the inpainted areas in MSI show blurry artifacts due to mismatched pixel correspondences when generating an MPI from such independently inpainted multi-view images and when generating MSI by MPI merging. Such disagreements over the multi-view images result in strong depth discontinuities, as evident in the depth channel. Per-layer inpainting results show stronger depth discontinuities, because the algorithm cannot maintain any connectivity across layers.

Comparison on Google dataset For the MSI of Google's dataset [7], we performed *SSD+FI* and *PL-PM*. We omit the other variants, since *SSD+FI* performed best on our original dataset (see also Section 4.4). Fig. 6 shows inpainting results in two scenes of the dataset. The subsets show the enlarged color and depth renderings of the same area. In the Beach Puppy scene, we removed the puppy standing on the reflective surface, i.e., the waves breaking on the shore. In the Flames scene, we wanted to remove the person and the flames, which occupy a large part of the scene, leaving some parts completely unobserved, as apparent in the mask.

While *PL-PM* delivers high-quality inpainting in the color channel, its restriction to a single layer corrupts the depth. Consequently, thin layers darken in the color domain when the fallback pixels are black, or, otherwise, present a wrong depth.

4.4 User study

Design. We designed a repeated measures within-subjects study to compare the quality of the inpainting of different inpainting methods. To that aim, we introduced the independent variable "inpainting method" with nine conditions: *Original* (The original MSI with no inpainting applied), *Masked* (MSI after the labeling performed), *SSD*, *SSD+I*, *SSD+F*, *SSD+FI*, *MV-CNN*, *MV-PS*, and *ML-PM*. We included *Original* to confirm that all participants understood the purpose of this study: Original should always receive the lowest score. As dependent variables, we collected ratings for inpainting results on a 7-point scale. All results were obtained by showing videos of a static MSI with a camera panning in a circle to enhance the motion parallax. Due to the scale ambiguity of our structure-from-motion method (COLMAP), the motion range was manually adjusted.

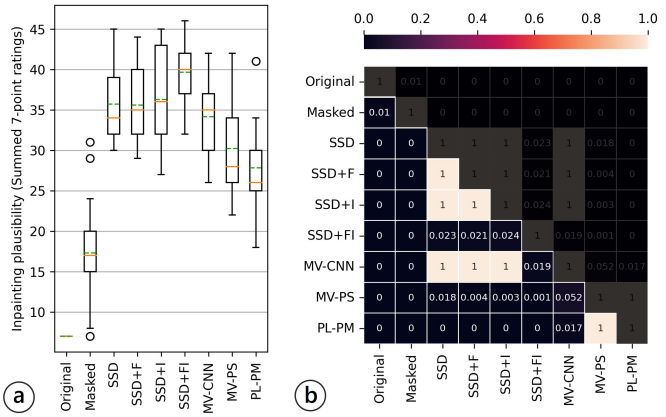

Fig. 8: User study results. (a) The boxplots (the orange and green bars represent medians and means) and (b) a significance map. The boxplots illustrate some important aspects of the collected data: *Original* obtained the lowest score, 1, as we expected. *Masked* follows *Original*. *SSD+FI* surpasses all other approaches. (b) The significance map summarizes all p values in the statistical analysis.

Task. We designed a task to rate inpainting results using the seven scenes in our dataset, including the scenes in Fig. 5. First, the participants were instructed to observe which part of a scene should be inpainted. We presented a static image with highlighted objects to be inpainted on the left and a video with a moving camera in the multi-layer image on the right. Then, we asked the participants to evaluate the overall quality of the inpainting method. We inquired on the plausibility of the inpainting, *"Does the video show the scene as if there were no such visual obstacles in the first place?"* The participants reported the rating on a 7-point Likert scale, where seven means most positive.

Apparatus. We used a web-based survey system, Microsoft Forms, to collect responses from people of different expertise and nationalities, after an email invitation. Participants were instructed to use at least a 12" screen to ensure reasonable viewing conditions.

Procedure. After receiving textual instructions and signing an informed consent form, the participants evaluated a series of inpainting videos. 18 participants (six females, age $\bar{X} = 29.30$, SD $= 5.90$) volunteered for the study. On a scale of one to five, where five means best, the mean self-rated experience of inpainting was 2.33 (SD $= 1.14$). The participants scored all cases in random order to avoid any learning effect. A session took approximately 38 minutes. With 18 participants, seven scenes, and nine inpainting methods, we collected a total of $18 \times 7 \times 9 = 1,134$ ratings.

Hypotheses. From the discussions in the previous sections 4.3 and 4.3, we expected that *SSD+FI* would receive the best score (**H1**). We also expected that *Masked* would score worst, as it always reveals the fallback pixels (**H2**), and that *PL-PM* would follow as second worst (**H3**). Moreover, we expected that our methods (*SSD*, *SSD+F*, and *SSD+I*) would outperform the multi-view (*MV-CNN* and *MV-PS*) and per-layer (*PL-PM*) methods, due to the disagreeing depths over the input multi-view images and inconsistent image inpainting over layers, respectively (**H4**).

Results. We evaluated summed scores over the scenes for each inpainting method. Therefore, the maximum score of a method was 49 (the highest score of 7×7 scenes). We confirmed that *Original* always received the lowest scores. We confirmed the normality but not sphericity on the collected score data excluding the *Original* condition. We therefore performed a Friedman test to examine the effect that the nine different inpainting methods had on the score. The results showed that the inpainting method led to statistically significant differences in rating ($\chi^2(8) = 119.63$, $W = 0.83$, $p = 0.00$).

Fig. 9: Multi-layer scene augmentation by inserting 3D objects in the Alexa Meade Exhibit scene [7] – best seen using red-cyan anaglyph glasses. This example shows the original multi-layer scene without and with 3D objects, and the inpainted scene without and with 3D objects. When the black poles are removed from the scene, the inserted artificial signboards appear more coherent in the scene.

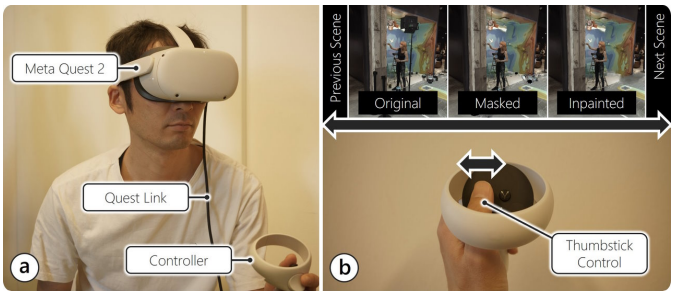

Fig. 10: VR viewer. (a) We implemented a viewer running on Unity with Meta Quest 2. (b) With the thumbstick control, the viewer can navigate different multi-layered scenes. Please refer to the supplemental video for the demonstration in action.

We, therefore, performed Wilcoxon signed-rank tests on the data. Fig.8 summarizes the results as (a) data boxplots and (b) a significance map. Notably, *SSD+FI* outperforms the other approaches, including our patch metric variants, the two multi-view approaches, and the per-layer approach. The median differences are 5.5 ($p < 0.05$), 5.5 ($p < 0.05$), 4.5 ($p < 0.05$), 5.0 ($p < 0.05$), 12.0 ($p < 0.01$), 13.0 ($p = 0.0$) in *SSD*, *SSD+F*, *SSD+I*, *MV-CNN*, *MV-PS*, and *PL-PM*, respectively. Therefore, **H1** is statistically supported. *Masked* has the worst score among the solutions ($p = 0.0$), and thus, **H2** is also supported. *PL-PM* is placed at the second worst in median but contends with *MV-PS* (median difference: 1.0($p = 1.0$)). Compared with *MV-CNN*, *PL-PM* shows its inferiority (median difference: 8.0 ($p < 0.05$)). Therefore, we conclude that **H3** is partly supported. We observe that our patch metric variants outperform *MV-PS* (median differences: 6.5 ($p < 0.05$), 6.5 ($p < 0.01$), and 7.5 ($p < 0.01$) for *SSD*, *SSD+F*, and *SSD+I*, respectively) and *PL-PM* (median differences: 7.5 ($p = 0.0$), 7.5 ($p = 0.0$), and 8.5 ($p = 0.0$) for *SSD*, *SSD+F*, and *SSD+I*, respectively) while there is no clear superiority over *MV-CNN* (median differences: 0.5 ($p = 1.0$) for *SSD*, *SSD+I*, and *SSD+F*). Consequently, **H4** is partly supported. Overall, the results provide us with statistical evidence for the discussions on the qualitative results in Sections 4.3 and 4.3.

4.5 VR viewing

We invited five additional participants to collect informal comments on the VR viewing setup. A system, which uses an NVIDIA GeForce RTX 3060, runs approximately at 72 Hz on Unity 2020.3.32f1 and allows viewers to switch between original, masked, and our *SSD+FI* results in each scene using the left controller (Fig. 10). All participants expressed positive impressions of the improved quality in inpainted results compared to those in a masked MSI. They also noted the stereoscopic effects were well preserved after the inpainting and that general immersive experiences in an MSI were promising. Two participants specifically commented that the reflections on the wet sandy beach in the Beach Puppy scene looked coherent and reasonable in the inpainted results. One participant who works in computer vision pointed out inconsistent patterns in the inpainted area on the tiling of the Flames scene.

5 Limitations and Future Work

While our multi-layer inpainting approach shows promising results in two multi-layer scene datasets, some points need to be addressed to further improve the quality.

3D labeling tool. The prototype of our labeling tool is based on manual labor and thus on the user experience with the tool. Automating this process would definitely increase overall usability. However, to the best of our knowledge, there is no labeling method specifically designed for multi-layer images of soft 3D scene representations. Solutions for 3D segmentation [10] could be used, but detecting segments of, for example, the ambiguous reflections of the puppy in the Beach Puppy scene (Fig. 6 top) seems challenging even for state-of-the-art segmentation methods.

As a single 3D object softly spreads out in multiple layers, it becomes challenging for humans to label parts of the object represented by pixels with low alpha values, which are almost invisible. Although such pixels often exist in so-called "stacked cards" artifacts, these artifacts tend to be unfamiliar to many users and difficult to clean. An interactive preview may help to understand the on-edit scenes. However, understanding which parts across layers correspond to an object in the rendered view is generally hard.

Lack of patch samples. The primary issue in exemplar-based inpainting approaches is quality degradation due to the limited number of patch available in the scene. Therefore, we implemented intensity compensation and flipping, so that more patch candidates are available during the patch search. While individual measures have a limited effect, their combination noticeably improves quality (Section 4.4). However, overly large inpainting areas inevitably lead to patch depletion. In such a case, inpainted areas appear artificial due to unnatural patterns or blurry appearance (Fig. 9).

Other problems are caused by image noise. CNN methods for multi-layer image generation [16, 70] incorporate depth ambiguities from duplicated pixels behind actual scene points. These ambiguities are mostly unnoticeable, because they are camouflaged by the alpha compositing during multi-layer image rendering. However, for inpainting, noise outside of the masked area may confuse the patch search, resulting in failure to close the masked regions. Therefore, denoising or otherwise cleaning multi-layer images can improve robustness.

For multi-layer images with wide viewing angle and strong distortions due to sphere geometry proxies, we suggest to take panoramic image specific treatments to further increase patch candidates [17].

Camera alignment along the horizon. For best results, one must ensure that the horizontal axis of the world coordinate system and the real-world horizon match. Misalignment of the horizontal axes results in pixel bands bent in a circle, affecting our warping function, f_{F}. Horizon misalignment rarely occurs in a dedicated camera rig, which is placed perpendicularly to the ground [7], but a dataset built from unstructured multi-view images places the world axis arbitrarily. For our original dataset, we manually aligned the horizon. Gravity-aware camera registration methods would mitigate this misalignment [30].

Multi-layer video inpainting. Applying our multi-layer image inpainting algorithm to temporally sequential multi-layer images would result in multi-layer video inpainting. However, for video inpainting, a different inpainting strategy would perform better than a straightforward extension of our approach. For example, spatio-temporal inpainting [65] searches patches not only in space, but also in time, benefiting from moving objects revealing parts of the backgrounds over time. Furthermore, multi-layer videos are likely to use different data structures, such as layered meshes [7].

Augmenting multi-layer scenes with 3D objects. After removing objects from a scene, one may add 3D objects to further enhance the multi-layer scene (Fig. 2d). We demonstrate this application using Unity 3D. Fig. 9 presents the original scene, augmented original scene, inpainted scene, and augmented scene after inpainting at the same camera pose, shown in red-cyan anaglyph stereo. After removing the black poles and cans, we placed two artificial signboards. The signboards are a conventional 3D polygon model. Since the multi-layer scene representation consists of layered 3D meshes, we are able to trivially render new 3D objects and layered image content in the same immersive scene.

Real-time metameric inpainting. The latest metameric inpainting [29] performs real-time inpainting with less noticeable filled-in pixels. Such a method could be integrated with our inpainting algorithm, for example, as a subsequent process for our masked rendering results. However, multi-layer images do not have explicit depth maps but instead rendered depth maps, which may require special consideration when using them as depth maps.

6 Conclusion

This paper presents a novel inpainting pipeline for multi-layer images. Our evaluations and user study demonstrate how our inpainting method outperforms single-layer inpainting algorithms on multi-view images. We explore the impacts of different patch metrics on multi-layer image inpainting. Multi-layer image inpainting opens up a new research direction towards editable VR photography.

Acknowledgments

This work was supported by the Austrian Science Fund FWF (grant no. P33634).

References

[1] Alex Yu and Sara Fridovich-Keil, M. Tancik, Q. Chen, B. Recht, and A. Kanazawa. Plenoxels: Radiance fields without neural networks, 2021. 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 *Proc. European Conference on Computer Vision (ECCV)*, pp. 441–459. Springer, 2020. 1, 2, 3

[3] S.-H. Baek, I. Choi, and M. H. Kim. Multiview image completion with space structure propagation. In *Proc. Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 488–496, 2016. 1, 2

[4] C. Barnes, E. Shechtman, A. Finkelstein, and D. B. Goldman. Patchmatch: A randomized correspondence algorithm for structural image editing. *ACM Transactions on Graphics (TOG)*, 28(3):24, 2009. 2, 3, 4, 5

[5] C. Barnes, E. Shechtman, D. B. Goldman, and A. Finkelstein. The generalized patchmatch correspondence algorithm. In *Proc. European Conference on Computer Vision (ECCV)*, pp. 29–43. Springer, 2010. 2, 3, 4, 5

[6] A. E. Beaton and J. W. Tukey. The fitting of power series, meaning polynomials, illustrated on band-spectroscopic data. *Technometrics*, 16(2):147–185, 1974. 4

[7] M. Broxton, J. Flynn, R. Overbeck, D. Erickson, P. Hedman, M. Duvall, J. Dourgarian, J. Busch, M. Whalen, and P. Debevec. Immersive light field video with a layered mesh representation. *ACM Transactions on Graphics (TOG)*, 39(4):86–1, 2020. 1, 2, 3, 5, 6, 7, 8, 9

[8] C. Buehler, M. Bosse, L. McMillan, S. Gortler, and M. Cohen. Unstructured lumigraph rendering. In *Proc. Conference on Computer Graphics and Interactive Techniques (SIGGRAPH)*, pp. 425–432, 2001. 2

[9] P. Buyssens, M. Daisy, D. Tschumperlé, and O. Lézoray. Exemplar-based inpainting: Technical review and new heuristics for better geometric reconstructions. *IEEE Trans. on Image Processing*, 24(6):1809–1824, 2015. 2

[10] Ö. Çiçek, A. Abdulkadir, S. S. Lienkamp, T. Brox, and O. Ronneberger. 3d u-net: Learning dense volumetric segmentation from sparse annotation. In S. Ourselin, L. Joskowicz, M. R. Sabuncu, G. Unal, and W. Wells, eds., *Proc. Medical Image Computing and Computer-Assisted Intervention (MICCAI)*, pp. 424–432. Springer International Publishing, Cham, 2016. 8

[11] A. Criminisi, P. Perez, and K. Toyama. Region filling and object removal by exemplar-based image inpainting. *IEEE Transactions on Image Processing (TIP)*, 13(9):1200–1212, September 2004. 2

[12] A. Dai, D. Ritchie, M. Bokeloh, S. Reed, J. Sturm, and M. Nießner. Scancomplete: Large-scale scene completion and semantic segmentation for 3d scans. In *Proc. Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018. 2

[13] A. Davis, M. Levoy, and F. Durand. Unstructured light fields. In *Computer Graphics Forum*, vol. 31, pp. 305–314. Wiley Online Library, 2012. 2

[14] P. Debevec, Y. Yu, and G. Borshukov. Efficient view-dependent image-based rendering with projective texture-mapping. In *Eurographics Workshop on Rendering Techniques*, pp. 105–116. Springer, 1998. 2

[15] C. Ebner, S. Mori, P. Mohr, Y. Peng, D. Schmalstieg, G. Wetzstein, and D. Kalkofen. Video see-through mixed reality with focus cues. *IEEE Transactions on Visualization and Computer Graphics (TVCG)*, 28(5):2256–2266, 2022. 2

[16] J. Flynn, M. Broxton, P. Debevec, M. DuVall, G. Fyffe, R. Overbeck, N. Snavely, and R. Tucker. Deepview: View synthesis with learned gradient descent. In *Proc. Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2367–2376, 2019. 1, 2, 9

[17] V. Gkitsas, V. Sterzentsenko, N. Zioulis, G. Albanis, and D. Zarpalas. Panodr: Spherical panorama diminished reality for indoor scenes. In *Proc. Conference on Computer Vision and Pattern Recognition Workshop (CVPRW)*, pp. 3716–3726, 2021. 9

[18] S. J. Gortler, R. Grzeszczuk, R. Szeliski, and M. F. Cohen. The lumigraph. In *Proc. Conference on Computer Graphics and Interactive Techniques (SIGGRAPH)*, pp. 43–54, 1996. 2

[19] B. Guenter, M. Finch, S. Drucker, D. Tan, and J. Snyder. Foveated 3d graphics. *ACM Transactions on Graphics (TOG)*, 31(6):1–10, 2012. 2

[20] A. Isaksen, L. McMillan, and S. J. Gortler. Dynamically reparameterized light fields. In *Proc. Conference on Computer Graphics and Interactive Techniques (SIGGRAPH)*, pp. 297–306, 2000. 2

[21] H. Ishiguro, M. Yamamoto, and S. Tsuji. Omni-directional stereo. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 14(02):257–262, 1992. 2

[22] R. Ishikawa, H. Saito, D. Kalkofen, and S. Mori. Multi-layer scene representation from composed focal stacks. *IEEE Transactions on Visualization and Computer Graphics (TVCG)*, 2023. 2

[23] C. Jambon, B. Kerbl, G. Kopanas, S. Diolatzis, T. Leimkühler, and G. Drettakis. Nerfshop: Interactive editing of neural radiance fields". *Proc. ACM Symposium on Interactive 3D Graphics and Games (I3D)*, 6(1), May 2023. 2

[24] V. Jampani, H. Chang, K. Sargent, A. Kar, R. Tucker, M. Krainin, D. Kaeser, W. T. Freeman, D. Salesin, B. Curless, et al. SLIDE: Single image 3d photography with soft layering and depth-aware inpainting. In *Proc. International Conference on Computer Vision (ICCV)*, pp. 12518–12527, 2021. 2

[25] A. Jarabo, B. Masia, A. Bousseau, F. Pellacini, and D. Gutierrez. How do people edit light fields. *ACM Transactions on Graphics (TOG)*, 33(4):4, 2014. 2

[26] N. Kawai, T. Sato, and N. Yokoya. Image inpainting considering brightness change and spatial locality of textures and its evaluation. In *Pacific-Rim Symposium on Image and Video Technology*, pp. 271–282. Springer, 2009. 3, 4

[27] N. Kawai and N. Yokoya. Image inpainting considering symmetric patterns. In *Proc. International Conference on Pattern Recognition (ICPR*, pp. 2744–2747. IEEE, 2012. 3, 4

[28] P. Kellnhofer, L. C. Jebe, A. Jones, R. Spicer, K. Pulli, and G. Wetzstein. Neural lumigraph rendering. In *Proc. Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 4287–4297, 2021. 2

[29] R. Kuffner dos Anjos, D. R. Walton, K. Aksit, S. Friston, D. Swapp, A. Steed, and T. Ritschel. Metameric inpainting for image warping. *IEEE Transactions on Visualization and Computer Graphics (TVCG)*, pp. 1–12, 2022. 9

[30] D. Kurz and S. Benhimane. Handheld augmented reality involving gravity measurements. *Computers & Graphics*, 36(7):866–883, 2012. 9

[31] M. Le Pendu, X. Jiang, and C. Guillemot. Light field inpainting propagation via low rank matrix completion. *IEEE Transactions on Image Processing (TIP)*, 27(4):1981–1993, 2018. 1, 2

[32] M. Levoy. Light fields and computational imaging. *Computer*, 39(8):46–55, 2006. 2

[33] M. Levoy and P. Hanrahan. Light field rendering. In *Proc. Conference on Computer Graphics and Interactive Techniques (SIGGRAPH)*, pp. 31–42, 1996. 2

[34] J. Li, Y. Liu, X. Yuan, C. Zhao, R. Siegwart, I. Reid, and C. Cadena. Depth based semantic scene completion with position importance aware loss. *IEEE Robotics and Automation Letters*, 5(1):219–226, 2019. 2

[35] J. Li, G. von Campe, A. Pepe, C. Gsaxner, E. Wang, X. Chen, U. Zefferer, M. Tödtling, M. Krall, H. Deutschmann, U. SchÃ¤fer, D. Schmalstieg, and J. Egger. Automatic skull defect restoration and cranial implant generation for cranioplasty. *Medical Image Analysis*, p. 102171, 2021. 2

[36] K.-E. Lin, Z. Xu, B. Mildenhall, P. P. Srinivasan, Y. Hold-Geoffroy, S. DiVerdi, Q. Sun, K. Sunkavalli, and R. Ramamoorthi. Deep multi depth panoramas for view synthesis. In *Proc. European Conference on Computer Vision (ECCV)*, pp. 328–344. Springer, 2020. 1, 2, 5

[37] R. Martin-Brualla, N. Radwan, M. S. Sajjadi, J. T. Barron, A. Dosovitskiy, and D. Duckworth. Nerf in the wild: Neural radiance fields for unconstrained photo collections. In *Proc. Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 7210–7219, 2021. 2

[38] B. Mildenhall, P. P. Srinivasan, R. Ortiz-Cayon, N. K. Kalantari, R. Ramamoorthi, R. Ng, and A. Kar. Local light field fusion: Practical view synthesis with prescriptive sampling guidelines. *ACM Transactions on Graphics (TOG)*, 38(4):1–14, 2019. 1, 2, 3, 5, 7

[39] B. Mildenhall, P. P. Srinivasan, M. Tancik, J. T. Barron, R. Ramamoorthi, and R. Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. In *Proc. European Conference on Computer Vision (ECCV)*, pp. 405–421. Springer, 2020. 2

[40] P. Mohr, S. Mori, T. Langlotz, B. H. Thomas, D. Schmalstieg, and D. Kalkofen. Mixed reality light fields for interactive remote assistance. In *Proc. ACM Conference on Human Factors in Computing Systems (CHI)*, pp. 1–12, 2020. 2

[41] S. Mori, O. Erat, W. Broll, H. Saito, D. Schmalstieg, and D. Kalkofen. Inpaintfusion: Incremental rgb-d inpainting for 3d scenes. *IEEE Transactions on Visualization and Computer Graphics (TVCG)*, 26(10):2994–3007, 2020. 1, 2, 4

[42] S. Mori, J. Herling, W. Broll, N. Kawai, H. Saito, D. Schmalstieg, and D. Kalkofen. 3d pixmix: Image inpainting in 3d environments. In *Proc. International Symposium on Mixed and Augmented Reality Adjunct (ISMAR-Adjunct)*, pp. 1–2. IEEE, 2018. 1, 2

[43] T. Müller, A. Evans, C. Schied, and A. Keller. Instant neural graphics primitives with a multiresolution hash encoding. *ACM Transactions on Graphics (TOG)*, 41(4):102:1–102:15, July 2022. 2

[44] R. S. Overbeck, D. Erickson, D. Evangelakos, M. Pharr, and P. Debevec. A system for acquiring, processing, and rendering panoramic light field stills for virtual reality. *ACM Transactions on Graphics (TOG)*, 37(6):1–15, 2018. 2

[45] S. Park, X. Guo, H. Shin, and H. Qin. Surface completion for shape and appearance. *The Visual Computer*, 22(3):168–180, 2006. 2

[46] D. Pathak, P. Krahenbuhl, J. Donahue, T. Darrell, and A. A. Efros. Context encoders: Feature learning by inpainting. In *Proc. Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2536–2544, 2016. 2, 4

[47] A. Patney, M. Salvi, J. Kim, A. Kaplanyan, C. Wyman, N. Benty, D. Luebke, and A. Lefohn. Towards foveated rendering for gaze-tracked virtual reality. *ACM Transactions on Graphics (TOG)*, 35(6):1–12, 2016. 2

[48] Y. Peng, S. Choi, N. Padmanaban, and G. Wetzstein. Neural holography with camera-in-the-loop training. *ACM Transactions on Graphics (TOG)*, 39(6):1–14, 2020. 2

[49] E. Penner and L. Zhang. Soft 3d reconstruction for view synthesis. *ACM Transactions on Graphics (TOG)*, 36(6):1–11, 2017. 1, 2, 5

[50] J. Philip and G. Drettakis. Plane-based multi-view inpainting for image-based rendering in large scenes. In *Proc. ACM Symposium on Interactive 3D Graphics and Games (I3D)*, 2018. 1

[51] T. Porter and T. Duff. Compositing digital images. In *Proc. Conference on Computer Graphics and Interactive Techniques (SIGGRAPH)*, pp. 253–259. Association for Computing Machinery, New York, NY, USA, 1984. 3

[52] M. Richard and M. Y.-S. Chang. Fast digital image inpainting. In *Proc. International Conference on Visualization, Imaging and Image Processing (VIIP*, pp. 106–107, 2001. 2

[53] J. L. Schönberger and J.-M. Frahm. Structure-from-motion revisited. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 5

[54] A. Serrano, I. Kim, Z. Chen, S. DiVerdi, D. Gutierrez, A. Hertzmann, and B. Masia. Motion parallax for 360 rgbd video. *IEEE Transactions on Visualization and Computer Graphics (TVCG)*, 25(5):1817–1827, 2019. 2

[55] J. Shade, S. Gortler, L.-w. He, and R. Szeliski. Layered depth images. In *Proc. Conference on Computer Graphics and Interactive Techniques (SIGGRAPH)*, pp. 231–242, 1998. 2

[56] M.-L. Shih, S.-Y. Su, J. Kopf, and J.-B. Huang. 3d photography using context-aware layered depth inpainting. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020. 2

[57] H.-Y. Shum and L.-W. He. Rendering with concentric mosaics. In *Proc. Conference on Computer Graphics and Interactive Techniques (SIGGRAPH)*, pp. 299–306, 1999. 2

[58] V. Sitzmann, S. Rezchikov, W. T. Freeman, J. B. Tenenbaum, and F. Durand. Light field networks: Neural scene representations with single-evaluation rendering. In *arXiv*, 2021. 2

[59] Y. Song, C. Yang, Z. Lin, X. Liu, Q. Huang, H. Li, and C.-C. J. Kuo. Contextual-based image inpainting: Infer, match, and translate. In *Proc. European Conference on Computer Vision (ECCV)*, pp. 3–19, 2018. 2

[60] P. P. Srinivasan, R. Tucker, J. T. Barron, R. Ramamoorthi, R. Ng, and N. Snavely. Pushing the boundaries of view extrapolation with multiplane images. In *Proc. Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 175–184, 2019. 1, 2, 3, 7

[61] R. Szeliski and P. Golland. Stereo matching with transparency and matting. *International Journal of Computer Vision (IJCV)*, 32(1):45–61, 1999. 1, 2

[62] J. Thatte and B. Girod. Towards perceptual evaluation of six degrees of freedom virtual reality rendering from stacked omnistereo representation. *Proc. Electronic Imaging (EI)*, 2018(5):352–1, 2018. 1

[63] T. Thonat, E. Shechtman, S. Paris, and G. Drettakis. Multi-view inpainting for image-based scene editing and rendering. In *Proc. International Conference on 3D Vision*, pp. 351–359. IEEE, 2016. 1

[64] V. Vaish, B. Wilburn, N. Joshi, and M. Levoy. Using plane+ parallax for calibrating dense camera arrays. In *Proc. Conference on Computer Vision and Pattern Recognition (CVPR)*, vol. 1, pp. I–I. IEEE, 2004. 2

[65] Y. Wexler, E. Shechtman, and M. Irani. Space-time completion of video. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 29(3):463–476, 2007. 9

[66] Z. Yi, Q. Tang, S. Azizi, D. Jang, and Z. Xu. Contextual residual aggregation for ultra high-resolution image inpainting. In *Proc. Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 7508–7517, 2020.

7

[67] J. Yu, Z. Lin, J. Yang, X. Shen, X. Lu, and T. S. Huang. Generative image inpainting with contextual attention. In *Proc. Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 5505–5514, 2018. 2

[68] J. Yu, Z. Lin, J. Yang, X. Shen, X. Lu, and T. S. Huang. Free-form image inpainting with gated convolution. In *Proc. International Conference on Computer Vision (ICCV)*, pp. 4471–4480, 2019. 5, 7

[69] P. Yu, X. Yang, and L. Chen. Parallel-friendly patch match based on jump flooding. In *Advances on Digital Television and Wireless Multimedia Communications*, pp. 15–21. Springer, Berlin, Heidelberg, 2012. 5

[70] T. Zhou, R. Tucker, J. Flynn, G. Fyffe, and N. Snavely. Stereo magnification: Learning view synthesis using multiplane images. *ACM Transactions on Graphics (TOG)*, 37(4), jul 2018. 1, 2, 9