Metameric Inpainting for Image Warping

Rafael Kuffner dos Anjos, David R. Walton, Kaan Akşit, Sebastian Friston, David Swapp, Anthony Steed and Tobias Ritschel

Abstract—*Image-warping*, a per-pixel deformation of one image into another, is an essential component in immersive visual experiences such as virtual reality or augmented reality. The primary issue with image warping is disocclusions, where occluded (and hence unknown) parts of the input image would be required to compose the output image. We introduce a new image warping method, *Metameric image inpainting* - an approach for hole-filling in real-time with foundations in human visual perception. Our method estimates image feature statistics of disoccluded regions from their neighbours. These statistics are inpainted and used to synthesise visuals in real-time that are less noticeable to study participants, particularly in peripheral vision. Our method offers speed improvements over the standard structured image inpainting methods while improving realism over colour-based inpainting such as push-pull. Hence, our work paves the way towards future applications such as depth image-based rendering, 6-DoF 360 rendering, and remote render-streaming.

Index Terms—Inpainting, warping, perception, real-time rendering

1 Introduction

THE quality requirements for computer-generated content have been increasing for many years, with no sign of slowing down. Meanwhile, immersive, mobile, and remote applications have gained popularity. These applications have either higher rendering requirements (e.g., high, constant frame-rate or stereo), run on less powerful devices, or have limited access to data.

Image warping is an operation that allows re-rendering frames from alternative viewpoints using present per-pixel motion or view information. Alternative viewpoints can be offset in space or time. Image warping plays a crucial role in enabling these novel applications through latency compensation, stereo view synthesis or temporal upsampling [28]. A problem that will inevitably arise during warping is the disocclusion of regions for which there is no content to warp. Filling these "holes" with perceptually inaccurate content reduces the perceived realism of the rendered scenes. Thus, *Inpainting* algorithms fill a region of unknown pixels with plausible content [3].

Our definition of *plausible* depends on the application, context, and viewing conditions. Ideally, we would be able to predict precisely the missing information (e.g., predicting a mouth or eye on a face with a missing piece). In practice though, it is sufficient that the approximation is adequate for the context. When inpainting video, one might be able to find the accurate information to inpaint from future or past frames [33], but this is not guaranteed, and only viable if the video completion operation is performed offline due to the complex nature of this task. Recently, this problem has been approached using neural networks [44, 51, 52], which are able to take surroundings into account when predicting the missing content. These neural network approaches have been used extensively in image restoration and completion applications. However, they are typically complex to control, many are not temporally coherent, and their execution times prohibit real-time applications.

In this paper we propose *metameric image inpainting*. In colourimetry, two colours are considered metamers if they have different spectral power distributions, but are perceived as the same. Unlike metamers in colourimetry, Freeman and Simoncelli [11] explore a different type of metamer: images that are considerably different in content but are perceived as the same. An excellent example of such metamers as explored by Freeman and Simoncelli [11] are *ventral metamers* (see also [16, 40, 41, 48]), which are pairs of images that are perceived identically by peripheral vision. To briefly summarise, different patches may be perceived as the same due to the similarity in image statistics, which are vital components of the visual system. Therefore, it does not matter what exactly is being inpainted into holes, it should just agree in the statistics with what would be there. Our main observation is that methods such as the classic push-pull algorithm [15] inpaint missing regions with low-frequency content only, which can lead to unconvincing results when the high-frequency statistics are not matched.

Our hypothesis is that inpainting a disoccluded region with visual metamers improves the plausibility of warped images compared to naïve inpainting algorithms. This is aligned with the physiology of human vision for two reasons: First, if inpainting happens in the periphery – the largest part of the image – it is known [12, 39, 48] that a metamer is perceived to be more similar to a reference than blur. Second, if the inpainting happens in the fovea, a metamer is favorable owing to the properties of typical applications for warping: in stereo view synthesis, fusion of regions without luminance patterns is harder or impossible, if contradicting [7]. In temporal upsampling or latency compensation applications, exposure of warped and inpainted frames is short, and at short exposures, the human visual system largely behaves as a texture discriminator [38], meaning that inpainting a disocclusion with content of a similar texture to the background will likely be sufficient.

Our implementation uses smooth image moments of steerable filters that can be calculated in real-time to analyze the content surrounding a disoccluded region, and synthesise a visual metamer to fill the missing part. The key to making this work is inpainting that stops at depth edges, and a one-pass extension to warping to fill

- *Rafael Kuffner dos Anjos is with the University of Leeds. r.kuffnerdosanjos@leeds.ac.uk*
- *All other authors are with University College London. https://vr.cs.ucl.ac.uk/research/pipelines/*

Manuscript received November 10, 2021

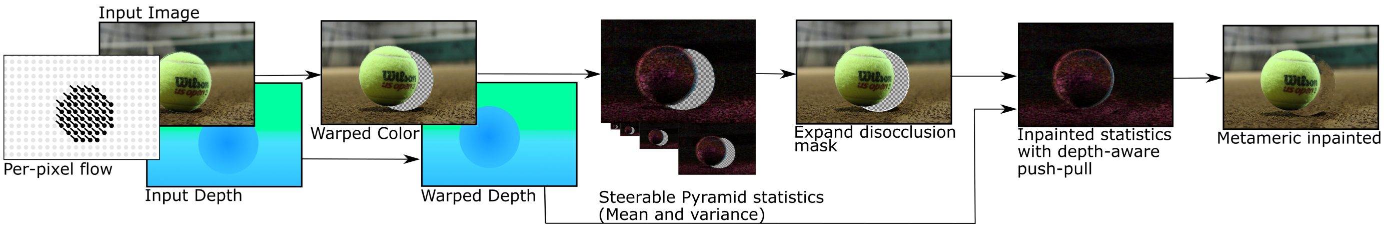


Fig. 1: Overview of our approach, including a warping step with consistent depth, a inpainting of moments and a metamerization step.

disocclusions with reliable depth values useful for edge-stopping. Technical contributions of our work include:

- A practical, parallel real-time method to fill disocclusion with patterns that share the visual statistics of their surroundings.
- A method to fill disocclusions with background depth in a single pass based on depth range partitioning.

2 Related Work

Our work combines two main themes in graphics: 3D image warping and plausible inpainting of image holes.

Warping composes a target image under some condition (view, time, light) by deforming an image made under another condition [28]. Common applications include temporal up-sampling [50], latency compensation [10] or synthesising stereo views from a single image [6]. A typical approach connects pixels at multiple resolution levels into polygons, which are then transformed and drawn into the new condition [6]. Alternatively, methods have been suggested to search for the source pixel to sample for the target image [32]. Several methods make use of more than one input image to be composed into a single target image [37, 43] or to store shading results into an atlas [31]. A primary difficulty with image warping is that some parts of the image under the target condition may not be observed in the source condition (*disocclusion*). Our approach is concerned with compensating for such missing areas with *inpainting*.

Inpainting seeks to fill missing parts of images ("holes") with plausible values. In our particular case, these holes are due to disocclusions of warping, although real-time inpainting has a range of other applications, including Diminished Reality (DR) [18, 30, 42].

A very simple inpainting method fills the colour values by a linear combination of neighbours, for example the popular push-pull method [15]. This approach is fast, but the resulting inpainted regions are strongly smoothed and lacking in higher frequency detail. More advanced methods exist, such as the often-used sequential approach [3], PatchMatch [1] and state-of-the-art methods using neural networks [44, 51, 52], but these are complex, non-GPU friendly and too computationally demanding for real-time, interactive applications. They are more suited to offline image-editing applications.

The inpainting task is slightly different for image warping that typically comes with access to a depth buffer [4, 14, 47], and where inpainting should handle the foreground and background differently. However, the depth is often not known for the holes, meaning that using it in a guided filter is a particularly hard challenge.

The idea of our inpainting is based on [48], which enables a fast method to extract spatially-localised statistics of filter responses [35] from a source image and apply them to a target image. Akin to texture synthesis, the resulting image is a "remix" of the input image that is perceived similarly i.e., they are metamers of each other [11]. While the original method has been applied to foveated rendering, where the statistics change according to the pooling of the ventral stream [39, 46] we here apply it to producing perceptually plausible patterns from a context. By induction, these patterns should be particularly effective when presented in the periphery of the viewer's vision, where the visual system only perceives pooled statistics, not details. Unlike other classic [9, 17, 24, 25, 34, 49] or learned texture synthesis work [13, 20, 21], this approach is localised in space (different textures in different places) and runs in real-time as it makes use of constant per-pixel time operations and moment maps [8].

Inpainting is now routinely used for novel-view synthesis, where stereo is estimated from a photo and warping in combination with inpainting enables changing the viewpoint [19]. These approaches rely on an intricate analysis of the input, a single static image, often involving executing one or multiple neural networks and optimizations that require in the order of seconds to produce high-quality results for varying views [23, 45, 54]. Our approach performs both the analysis of a changing input and the synthesis of an output at high quality and at high speed.

3 Real-time Warping with Plausible Disocclusions

Overview Our approach computes a warped RGB map without holes from an RGBZ map and a 2D flow map input as summarised in Fig. 1. First, we perform a modified warping operation that provides three results: the warped RGB map with holes, a warped Z image with background depth in disoccluded areas, and a binary disocclusion map (Sec. 3.1). Second, we calculate statistics of visual features across the unoccluded areas of the RGB map (Sec. 3.2). Third, we inpaint the disoccluded region with the statistics using a depth-aware push-pull (Sec. 3.3). Finally, a RGB realization of the statistics is computed to fill the disocclusions (Sec. 3.4). We will detail all four steps next.

3.1 Warping with Background Depth in Disocclusions

Our inpainting requires a specific warping operation to produce (1) an RGB map; (2) a binary occlusion map; and (3) a depth map in which disoccluded pixels have the depth value of the the background. The benefit of having background depth will be explained in Sec. 3.3.2, but it is intuitive to assume that disoccluded regions would have background depth and we want to inpaint from background to background and not from foreground to foreground.

Classic warping will provide (1) and (2), but not (3), which can be surprisingly hard to do. A naïve approach to get background depth is to apply push-pull [15] to the depth buffer. Unfortunately, this would create a smooth gradient of depth instead of the background depth. What we need instead is strictly the background, as we want the hole to be filled with a metamer that shares

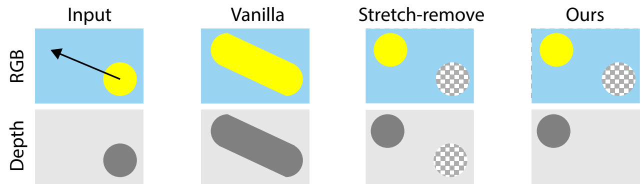


Fig. 2: Three ways to warp an input (first column) RGB image (top row) in conjunction with depth (bottom row): Drawing all pixel quads will "smear" the object across the image, producing neither correct depth nor a disocclusion mask (second column). Removing such stretched quads (third column) will avoid this issue and result in an occlusion mask but undefined holes in depth. Our approach fills holes with background depth (fourth column).

the statistics with only the background. Unfortunately, existing approaches to account for depth in push-pull [29] are not applicable here either, as they do not guarantee background, but close holes e.g., due to point rendering or foreground noise.

Instead of fixing depth post-hoc from an already-warped image, we suggest to address this ab-initio on the level of the warping. The idea is as follows (Fig. 3): when warping, neighbouring pixels are drawn as quads [6, 28, 37]. When a quad stretches more than a threshold it means it connects foreground and background. We call such a quad to be *stretched*. Drawing them, a circle warped on top of a plane would leave an unwanted "trail" (Fig. 2, second column). Hence, stretched quads are typically discarded in previous work (Fig. 2, third column). Our idea is not to eliminate, but to keep them in a special way.

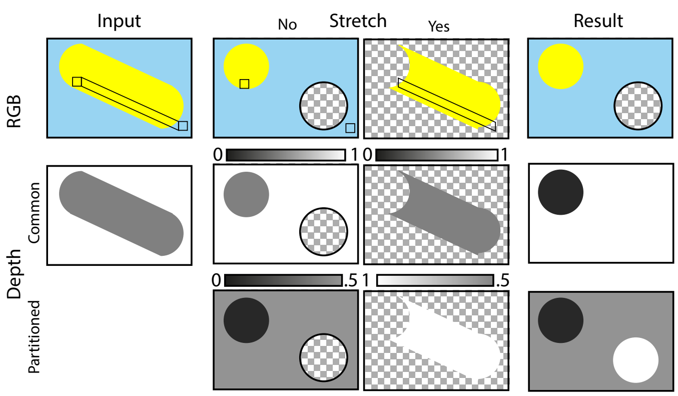


Fig. 3: Combining stretched and non-stretched quads: The first column shows the warped primitives in the scene from Fig. 2. The second and third column split the primitives between stretched and non-stretched ones. Both are drawn to the same colours and depth buffer, but with altered depth values. Three particular quads are indicated by the black lines in the image - two were not stretched by the warping, and the one connecting them has been stretched. The first row shows colour, the second row conventional depth values and the third row depths using our partitioning. The last column shows the result of the draw operations.

First, we note that *the minimum of the depth of all four vertices of a stretched quad is an approximation of the background depth*. Hence, we keep the stretched quad, but draw it in a special way as to only fill the hole with that minimal depth and leave all non-hole pixels unchanged. We do so by disabling interpolation of depths for stretched quads, writing the minimum depth of the four vertices at all pixels in the quad.

However, we still need to ensure the stretched quads are only rendered into disoccluded regions, and encode the disocclusion map in some way. We achieve both goals at once by *re-partitioning* our depth range. The depth at each pixel d is replaced by the re-partitioned depth d_r according to the following rule:

$$d_r = \begin{cases} 0.5d & \text{if quad is not stretched} \\ 1-0.5d & \text{if quad is stretched} \end{cases} \quad (1)$$

This maps all depths from non-stretched quads to the range $[0, 0.5]$ and all depths from stretched quads to the range $(0.5, 1]$, also flipping them in the process (i.e. 0.5 represents the greatest possible depth, and 1.0 the smallest). Note this implicitly encodes the disocclusion information in the depth map - if a pixel has a depth greater than 0.5, it belongs to a stretched quad, and is thus in a disoccluded region.

The remapping also means that stretched quads have greater depth values than non-stretched quads, and will always fail the depth test where a non-stretched quad is present. This means they will only be drawn into disoccluded regions.

In the event that two stretched quads overlap in a disoccluded region, since the depths are flipped, the quad with the greater raw depth value d will be drawn. This is desirable as our goal in the disoccluded regions is to render the surrounding background depths, and as such it makes sense to pick the most distant depth value in these cases.

For the purpose of the depth-aware inpainting, the depth values can be un-partitioned and mapped back to the usual original range.

3.2 Features of an incomplete image

We calculate a steerable pyramid [12] of an input image I in a decorrelated colour-space (YCbCr) [36], which estimates frequency responses at different scales and perceptual channels, mimicking the behaviour of the human visual system. Steerable pyramids apply a pair of direction sensitive filters to each level in the MIP map of I. The response to an orientation is a linear combination of the two main filter directions. Applying multiple pyramids at different orientations will deconstruct the image into frequencies at different orientations and scales, similar to a two-dimensional Fourier transform. Our pyramids are produced in real-time by convolving the source image with a set of small spatial filters, following [48]. We produce steerable pyramids for two orientations.

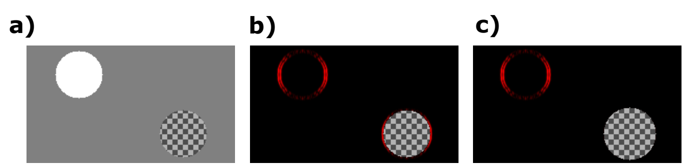


Fig. 4: Applying a steerable pyramid filter to an image with missing (disoccluded) regions can produce false responses. Disoccluded regions are shown as checkerboards. a) Input image. b) Horizontal filter response amplitudes show in red - note false responses around disocclusion. c) Dilating the disoccluded region to remove false responses.

However, if not treated specially, the missing regions in the input images would produce spurious frequency responses in the steerable pyramid, as the steerable filters would capture the change from background content to blank pixels, as shown in Fig. 4. The preferred way to handle undefined pixels are normalised convolutions [22]. These simply sum the product of weights with the alpha mask and divide (normalise) the convolution result by this value. We will indeed use such techniques to apply convex filters for inpainting statistics. Unfortunately, the feature detection filters in a steerable filter pyramid, akin to oriented edge filters, are concave, and normalised convolutions are not valid for concave filters. In fact, since the sum of the weights of an oriented steerable pyramid filter is zero, normalising such a filter is not in general mathematically well-defined.

Thus, after calculating the steerable pyramid of our input image, we expand the disoccluded region by the radius of our kernel K, treating its boundary as an unknown region, given that the filter responses there are unreliable. This is achieved by applying a morphological dilation operation to the disocclusion mask after filtering. Note, that the amount of dilation required is different for every level, as it depends on resolution. The dilated masks are only used where necessary, for performing convolutions with concave kernels - for other applications we use the original disocclusion masks.

3.3 Inpainting Statistics

Inpainting is performed for every level of the feature statistics pyramid independently. It maps a map of feature activations (explained in the previous Sec. 3.2) with holes to a map of feature activations statistics without holes. Two key aspects enable this, technically: very simple and compact moment descriptors (Sec. 3.3.1) and their edge-stopping inpainting (Sec. 3.3.2). We will discuss both, next.

3.3.1 Weighted Moments

We recall that [48] are creating smooth maps of moments (means $\mathbb{E}$ and variances $\mathbb{V}$) of feature responses X. As $\mathbb{V}[X] = \mathbb{E}[X]^2 - \mathbb{E}[X^2]$, in their task it is enough to blur feature maps X, as well as feature-square maps X^2 to compute the first two moments. The same works for inpainting, as any operator $\mathbb{E}_O$ that is a weighted mean (i.e., linear, positive-weighted, partition of unity) will also induce a weighted variance $\mathbb{V}_O[X] = O[X]^2 - O[X^2]$. Now, push-pull [15] itself is such a convex operator. Recall, that push-pull performs two passes: the first (pull) reduces resolution, averaging only valid values. The second (push) increases resolution again, replacing undefined pixels by blurry versions from a coarser resolution. Doing so, blur weights might vary spatially, even depend on context, but in the end they are positive weights, summing to one, multiplied with pixel values (be it pixel colour features or their squares), and therefore, push is also a convex operation. Hence, simply applying push-pull pp to the feature map X and squares-of-features map X^2 produces two other per-pixel maps $\mathrm{pp}(X)$ and $\mathrm{pp}(X^2)$ from which we read the two moments mean and variance $\mathrm{pp}(X)^2 - \mathrm{pp}(X^2)$, all in constant time per pixel and parallel.

The original push-pull algorithm uses normalised convolutions [22] in which a reduction of several input pixels into one output pixel will make that output pixel entirely valid as soon as any of the input pixels is valid. This is because the normalised convolution divides by the sum of the weights, except for the case where the divisor is zero, in which case the output remains undefined. We found this to be less temporally stable and use the following modification. Instead of eagerly making pixels valid as soon as possible, we track also partial weights when pulling. Doing so, pixels become valid more slowly, hence later in the pyramid, and so the result becomes spatially more blurry. Note that this blur is in the statistics domain, so the metamer realization still has all frequencies, just that their statistics change more slowly over space. This again leads to overblurring. To adjust temporal stability and spatial locality, we suggest applying a non-linearity to the alphas after each normalised convolution by raising them to a power, γ. For $\gamma = 1$, we have maximal temporal stability but spatial blur. For $\gamma = 0$ we would have the original push-pull with good locality but flicker. We present all our results for a compromise at $\gamma = .5$.

3.3.2 Edge-stopping

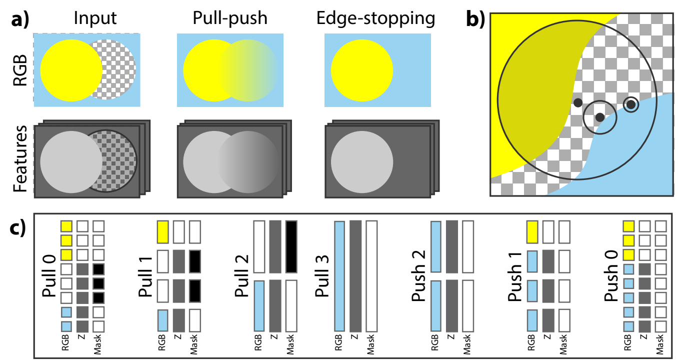


Fig. 5: **a)** Starting from an RGB (top) or feature map (Bottom) input (first column), common inpainting will blend between foreground and background areas (second column), while it should stop at edges (third column). **b)** The desired behavior for three points is to fetch information from circles just large enough to have enough valid information, to ignore undefined values and to ignore foreground values (darker yellow areas in the large circle). **c)** Our depth-aware push-pull for an 8-pixel 1D image.

Inpainting will however have a problem with foreground objects. For RGB images (top in Fig. 5, a), same as for the moments we use (bottom in Fig. 5, a), there will be an unwanted gradient between foreground appearance and background appearance as seen in the second column. The third column shows the desired behavior: inpainting the background. While we do not inpaint colours, but moments, the problem –and solution– is the same.

We make use of the fact that the warping has marked holes but also is filling them consistently with background depth (see section Sec. 3.1). The assumption is that disoccluded pixels would rather share statistics with the background than they would share with the foreground. This is not universally true, but a heuristic. It would be true for objects translating under an orthographic camera in front of a planar background. When the object rotates, it would disocclude parts of itself, which should belong to the foreground, an effect we do not model. Under perspective, even without rotation, foreground parts unobserved in the original view might become visible as well. In both cases, our approach would allocate them to background, shrinking the foreground object.

We adapt the push-pull to account for guidance by this depth map as seen in Fig. 5, b: to fill a value, we pull from a region just large enough to build statistics, but when doing so we ignore the undefined pixel, as well as pixels belonging to the foreground, here, yellow.

This is implemented as explained in Fig. 5, c. In the pull phase, we consider always 2×2 pixels being combined into one. In conventional push-pull, this is done by averaging all valid pixels in each block of four. We instead first find the minimal depth for the four-block. We then average those valid pixel values with depths within a set threshold of the minimal depth in the block.

Thus, moments from foreground objects never pollute background objects. Pseudo-code of both steps is given in Alg. 1.

Note that the PUSH procedure in Alg. 1 operates on two adjacent levels of the MIP pyramid, one high-resolution and one 4x lower resolution. The inputs to the procedure are the colour and validity values sampled from these two levels, as well as the parameter γ that controls the temporal stability of the output.

Algorithm 1 Pull and push step of metameric inpainting.

```
procedure PULL(colours[4], depths[4], validity[4])
    minDepth ← min(depths)
    for i ∈ [1,4] do
        if depths[i] - minDepth > threshold then
            validity[i] ← 0
        end if
    end for
    outcolour ← mean(colours × validity)
    outValidity ← mean(validity)
    outDepth ← minDepth
    return outcolour, outValidity, outDepth
end procedure
procedure PUSH(locolour, hicolour, hiValidity, γ)
    hiValidity ← pow(hiValidity, γ)
    return mix(hicolour, locolour, hiValidity)
end procedure
```

We note that whilst [29] also take depth into account in their pull phase, their goal is different. They inpaint in a surfel-based rendering setting, and attempt to avoid using background surfels visible in the gaps between foreground surfels. As such their depth test is reversed compared to ours; that is, they only draw from locations close to the maximal depth (closest to the camera).

3.4 Synthesis for hole-filling

Finally, we can use the statistics to synthesise content in the missing region, similarly to [48]. Given the statistics (μ, σ) of each component i, j of a steerable pyramid of l levels and b orientations, and a noise function $\xi_{i,j}$ the result is

$$r[x] = \mu_l + \sum_{i=0}^{l-1} \sum_{j=0}^{b-1} \mu_{i,j}[x] + \xi_{i,j}[x] \cdot \sigma_{i,j}[x]$$

where μ_l represents the residual lowpass of the steerable pyramid. The noise function $\xi_{i,j}$ filters white noise with the same steerable filters used to construct the i, j component of the pyramid, and scales it to a {-1,1} interval, allowing it to be shaped to fit the distribution described by $\mu_{i,j}, \sigma_{i,j}$. The other pixels can be copied from the input image, speeding up the process in the GPU.

3.4.1 Avoiding the Screen-door Effect

Use of a static noise function ξ in the synthesis process can lead to a visual artefact where background objects move, but noise remains static. We here refer to this artefact as the screen-door effect, by analogy with the similar artefact seen in VR headsets [2]. Since this artefact cannot be communicated in static images, we encourage readers to view our included video.

This effect can be mitigated by modifying the location at which the noise function ξ is sampled - i.e. at a screen location (x, y), we sample $\xi(x + \delta x, y + \delta y)$ where $(\delta x, \delta y)$ are the motion of the pixel at (x, y) since the last rendered frame. Since we inpaint disoccluded regions, the motion $(\delta x, \delta y)$ may not be known and must be estimated.

When warping using a motion field, we can also warp the motion field and apply the same depth-aware inpainting process used in Sec. 3.3.2 to estimate motion in the disoccluded regions. At each successive inpainted frame the sampling locations are iteratively moved along the motion field.

When warping using a 6DoF camera transform T (to inpaint 360 video for example) we make use of the inpainted depths to determine an appropriate sampling location $P \circ T \circ P^{-1}(x, y, z)$, where P is the camera projection function.

4 Results

Here, we provide results from our implementation for *metameric image inpainting*. We implemented our inpainting approach in Unity, which was also used to render 3D scenes to provide input for the approach. All results reported here use four steerable pyramid levels, with two orientations and 5×5 kernels, computed at a resolution of 1024×1024 unless said otherwise.

To provide a fair evaluation of our method, we also compare our method with state-of-the-art literature. Our comparison includes a naïve approach and a deep learning-based approach.

Naïve approach. The chosen method for the naïve approach is an algorithm called image-space reconstruction using push-pull interpolation [15]. Their algorithm consists of a pull phase and a subsequent push phase. The pull phase computes an image pyramid of a visual by reducing the image size with a factor of two at each step in the image pyramid. Down-sampling averages all valid pixels in each 2×2 pixel block of the image. In the push phase, pixels at each level interpolate the missing pixels in the original visual. We implemented the work by push-pull to derive results for the naïve approach (Fig. 6).

Deep learning based approaches. The image inpainting problem has garnered significant interest in the machine learning communities in recent years. We compare our approach to three deep-learning based inpainting methods, [44, 51, 52] Both [52] and [51] are image-based approaches, which accept an image with a binary mask, and inpaint the masked portions of the image. [44] instead reconstructs and completes a 3D mesh of the scene, which is then rendered to produce an output image or video sequence. Note that we do not compare quantitatively to [44]; this is because in practice their reconstructed mesh did not perfectly match the original geometry, resulting in high LPIPS errors that did not reflect the visual quality of the output. For all methods we use trained models provided by the authors. Results are shown in Tbl. 1.

Image quality Metamerised images do not have a common standard for image quality measurement purposes.

Comparing images with metamerised versions of the same images is not straightforward. Nevertheless, we applied LPIPS [53] to measure the difference of several methods.

TABLE 1: LPIPS image error for different scenes and methods.

	Our	PP	[52]	[51]
Castle	**.037**	.050	.038	.044
Castle2	**.025**	.033	.025	.030
Garden	**.017**	.023	.017	.023
Shed	**.012**	.016	.013	.018
Skate	.017	.015	**.013**	.015
Tunnel	**.009**	.013	.010	.013
	.019	.025	**.019**	.024

The methods were compared on a series of photogrammetric reconstructions of real scenes, mimicking the natural images used to train the neural-network approaches. A breakdown is seen in Tbl. 1. In each case the LPIPS values are average results over a short 120-frame sequence rendered with each scene. LPIPS losses were computed over the disoccluded regions only, in order to prevent any small differences between warped and ground truth pixel values affecting the loss (this was achieved by setting pixels outside disoccluded regions to equal those in the ground truth images).

We additionally compared the results of each method to the ground truth under the FovVideoVDP metric [27]. This is a perceptual metric of video quality, and tests for artefacts such as noise, or temporal flickering. This metric requires a model of the display used; for these tests, we used the "standard_fhd" model provided by the authors. Results are given as Just Objectionable Difference (JOD) values, which range from 0 to 10 (greater is better).

TABLE 2: JOD values under [27] for different scenes and methods.

	Our	PP	[52]	[51]
Castle	**6.91**	6.60	6.81	6.34
Castle2	**7.41**	7.27	7.35	6.76
Garden	**7.75**	7.66	7.71	6.96
Shed	**7.93**	7.85	7.84	6.86
Skate	**8.02**	7.84	7.95	7.21
Tunnel	**7.99**	7.77	7.66	7.94
	7.67	7.50	7.56	6.85

Speed We evaluate the performance of our inpainting implementation discussed in Sec. 3 compared to the classic push pull work by Gortler et al. [15] and a neural network-based inpainting approach by Yu et al. [52] (Tbl. 3). These results were obtained on a machine using an NVIDIA RTX 2070 GPU and an AMD Ryzen 3700X processor. We note that both push-pull and our own approach are more than an order of magnitude faster than the neural network approaches, and thus far better suited to real-time applications. It is challenging to directly compare to [44], as this method generates an inpainted 3D mesh through a computationally expensive process taking several minutes, but this mesh can then be rendered at interactive rates. However we note that any significant change to scene geometry or viewpoint would necessitate regenerating this mesh, making it unsuited to interactive 3D applications. By comparison to push-pull, our approach is roughly six times slower, owing mainly to the need to inpaint multiple pyramid levels rather than a single frame.

TABLE 3: Compute time.

	Res.	Time
Ours (5 lev.)	512^2	6.58 ms
	1024^2	17.86 ms
	2048^2	59.52 ms
Ours (3 lev.)	512^2	5.75 ms
	1024^2	16.56 ms
	2048^2	57.80 ms
Push-pull	512^2	1.47 ms
	1024^2	2.56 ms
	2048^2	8.70 ms
Neural net.		>1 s

We also compared our modified warping approach described in Sec. 3.1 to the naïve approach of discarding overly stretched triangles (rather than rendering them to produce estimates of the background depth).

Both approaches were implemented in shader code within the Unity game engine. In practice, the runtime of both approaches was identical. In principle the naïve approach discards triangles, reducing the rendering cost. However, only a very small proportion of the total triangles are affected, and this did not have a measurable impact on frame rate, even at high resolutions.

5 User Study

We conducted a user study to validate our hypothesis outlined in Sec. 3; that metameric image inpainting can be perceived as a closer approximation of a complete image than colour-based inpainting when both are visualised for a short amount of time. A total of $N = 11$ participants were recruited to carry out the experiment using a desktop-based Unity3D application. All of the experiments were carried out using the same screen and viewing distance to ensure comparable conditions. We compared having a ground truth video sequence to warping and inpainting using: push-pull interpolation [15], and metameric inpainting. To the best of our knowledge, there is no published neural network-based approach that works in real time that could be included in this comparison for the video resolution used in our study (2048×2048).

Protocol Participants were shown pairs of videos, time-divided, for 3 seconds each with a randomised display order. Videos were presented to users on a 27" FHD monitor, placed approximately 70cm away from the participants. All participants used the same display setup. Videos contained circular motion parallel to the image plane, revealing small (S) or large (L) disoccluded regions to be inpainted with each method. We included these variants to evaluate if the size of disocclusion had any effect on the success of each method. So for a total of six method-combinations (Reference, PP, Ours, with small and large disocclusions, only comparing within same size), users were shown six example scenes with two repeats, for a total of seventy-two decisions per participant. Participants were asked to choose which image they preferred from each pair (2AFC). Subjects were primed to consider "artifacts" and "overall quality".

Analysis Fig. 7 summarises preferences as probabilities for each combination. For each pair we perform a binomial test to check significance compared to chance. In all cases participants distinguished the inpainted stimuli from reference, though with a stronger effect size for PP compared to Ours. We find that our approach is preferred over PP in both small and large disocclusions with significant effects. This verifies our hypothesis that our approach produces inpainted content that is perceived as more plausible than previous work. Moreover, during post-experience interviews, subjects mentioned that metameric inpainting performed best when they were not looking directly at the inpainted regions, i.e. when happening in the periphery. The next section will discuss this in further detail, and how this can be applied to real use-case scenarios.

Foveated display application Metameric inpainting is best suited for peripheral vision, where the HVS is challenged to tell metamers from a reference [12, 48]. When we use them in the fovea as well, that is due to the lack of any better alternative, but we do not claim that they are perceived equivalently to a reference, only better than push-pull would be.

Foveated displays such as the Varjo XR-3, however, very well fit the metameric assumptions. They combine two displays, one with a high pixel density to be shown to the fovea and one with a lower pixel density shown to the periphery. These two displays are combined optically. When we reduce the image compute frequency of the peripheral display (for example from 90 to 30 Hz) we can use warping-based temporal upsampling [5, 6] with metameric inpainting to go back to 90 Hz. The foveal display keeps the

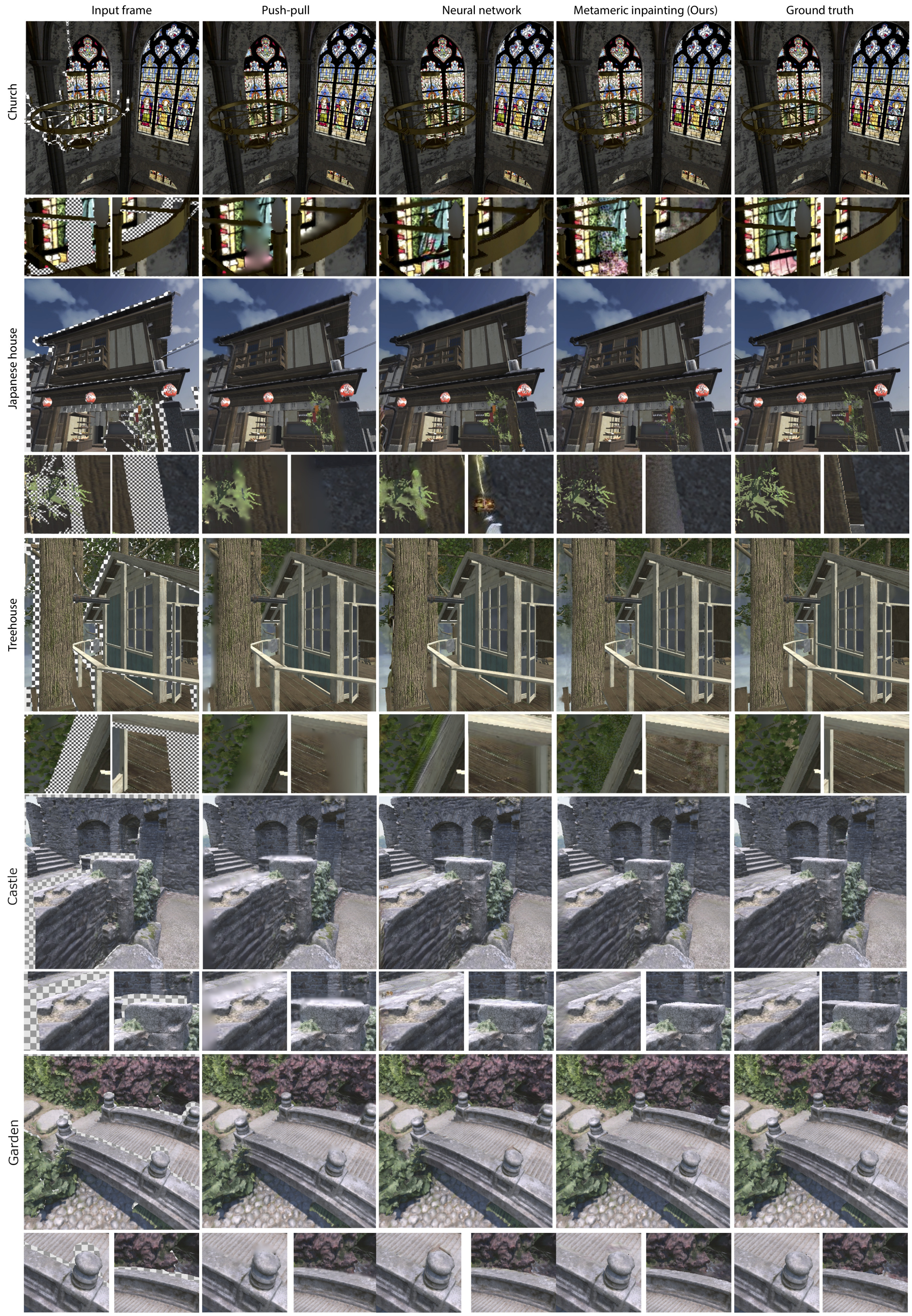


Fig. 6: Comparison of our metameric image inpainting method with push-pull [15] and a deep learning-based approach [52]. The first row shows the warped frame with a checkerboard to reveal disocclusions. Columns two, three and four are push-pull, NN and our method while the last column shows the reference of the target frame. Overall, our approach fares equally well or better than a NN while being two orders of magnitude faster. Please see the text in Sec. 6 for a detailed discussion.

[Scenes created by `aurelien_martel@SketchFab`, `noxfcna@sketchfab`, `artfletch@sketchfab`]

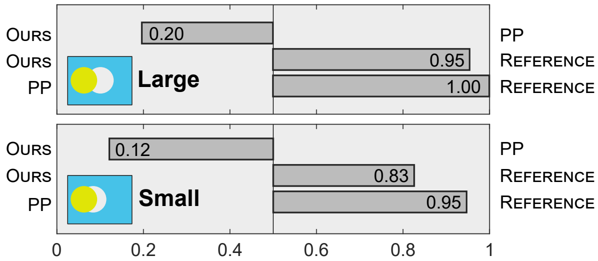


Fig. 7: Preferences as proportions for different forced binary choices between different treatments. All statements significant to $p < .0001$.

original 90 Hz. Like this, central vision is unaffected and the periphery sees a metamer it cannot distinguish from the reference.

We simulate appearance on a Varjo XR-3 in Fig. 9. We see that fixating the image center, a metameric warping is similar to the reference, while it is not for push-pull which appears blurry.

A limitation of this approach is that a foveated display will always show an optically band-limited version of the metamer, and hence can never fully match the reference in the periphery. Still, the frequency range present is sufficient to outperform push-pull.

6 Discussion

Our user study confirmed our hypothesis that metameric inpainting produces more plausible inpainted content than pull-push. This section will discuss the results of our quantitative comparison to other approaches, and show some specific examples.

In the quantitative comparison in Tbl. 1, our approach outperforms the others on all but one of the compared scenes. This is despite our method being more than an order of magnitude faster than the deep learning-based methods. As might be expected, methods [51, 52] suffer from flickering in the output videos, as they have no mechanism to enforce temporal consistency. This is reflected in their lower scores in Tbl. 2. [44] produced results with much better temporal consistency, but occasionally inaccurate geometry would be produced in the inpainted regions, harming the perceived quality of the results. Full results and videos are included in the supplemental material.

A comparison between our approach and naïve inpainting can be seen in Fig. 8. In both scenes, metameric inpainting is able to fill in the disoccluded region with plausible texture content that matches its surroundings, while not introducing unrealistic artifacts. Notably, our approach produces sharper outlines on foreground objects, and specially on the example to the right, is able to closely simulate the textured background. These examples also demonstrate how when located in the periphery, our approach is less noticeable than PP.

Fig. 6 shows an in-depth comparison between our approach and the proposed alternatives. Here we compare to [52], the deep learning approach that performed best in the quantitative comparison. Our approach does not distort the shape of foreground objects when inpainting background. On the Treehouse example, we can see the PP approach and [52] distorting the shape of the tree and wood beams, while ours preserves it. Similarly on the church, with the chandelier beams being distorted by these approaches. When comparing only to the PP approach, the teaser figure shows the flowers bleeding into the background, and both examples on Fig. 8 show similar foreground distortion effects. While [52] was able to better predict the wood texture on the treehouse, and create a more plausible result on the church, the results produced by our metameric inpainting are plausible synthesised textures, blending well with the environment and approximating the ground truth. A similar effect can be seen in Figure Fig. 6, b, with the content disoccluded by the pillar, and with the background of Fig. 8 d. The Japanese House scene shows an example of a failure case of [52], which predicted nonexistent objects in the disoccluded region. Our approach is able to produce correct textures for the wall section behind the pillar, with the higher frequency content being more in line with the reference than push-pull.

Limitations Our approach for temporal stability addresses the locality issue of push pull. However, new content being revealed as the size of disocclusions increases will inevitably introduce sudden changes in the calculated statistics, and the inpainted content. However, this limitation is only visible in large disocclusions, which are not the typical use cases discussed in this paper, or the highlighted applications. Even so, our approach was still found to be better than pull-push on large disocclusions. However, addressing these limitations would allow more freedom of movement in applications such as 6-DoF for 360 content or free viewpoint video for lumigraphs.

As seen in Fig. 10, we are not able to address the limitation of push pull of not being able to reproduce sharp edges in the disoccluded region, even if we correctly reproduce nearby textured patterns. Such scenarios are able to be addressed in offline methods (e.g. neural network approaches), and should be investigated for real-time in future work.

Finally, warping itself is subject to a number of limitations that cannot be overcome by our method such as handling of anti-aliased edges, motion blur or depth-of-field. We note, however, that anti-aliasing can be applied to the output of our approach, for example by rendering at a higher resolution and downsampling, or by applying any post-processing anti-aliasing approach such as Fast Approximate Anti-Aliasing [26]. Other post-processing effects (e.g. depth-based fog) could also be added at this stage.

7 Conclusions

We have proposed a method to combine the speed of classic RGB push-pull inpainting [15] with the quality of structured inpainting [3]. The neurophysiology of human perception inspires our proposal, which postulates the visual system to operate on statistics of features [48]. Hence, holes should not be filled with colours that agree with their surroundings, but with a pattern with the same statistics. Our approach provides a practical method to do so.

We inherit the typical limitations of warping, struggling with anti-aliasing, specular shading and transparent objects. Also, our approach is slower than push-pull on RGB, given that more calculations are needed. Usefulness depends on the application, the size of the warp (and hence the size of the holes), and the cost of rendering. Future work could combine foveated rendering and foveated inpainting.

We believe various applications such as depth image-based rendering, 6-DoF rendering, and remote rendering-streaming can take advantage of our method, which combines high-performance computation and perceptual principles.

Acknowledgments

This work was funded in part by the EPSRC/UKRI project EP/T01346X/1.

Fig. 8: Comparison between push-pull [15] (top) and ours (bottom) on a variety of additional scenes. [Scenes created by bastienBGR@SketchFab and aurelien_martel@SketchFab]

References

[1] Connelly Barnes, Eli Shechtman, Adam Finkelstein, and Dan B Goldman. Patchmatch: A randomized correspondence algorithm for structural image editing. *ACM Trans. Graph.*, 28(3):24, 2009.

[2] Joung-min Cho, Young-do Kim, Song Hee Jung, Hyunchang Shin, and Taesung Kim. 78-4: Screen door effect mitigation and its quantitative evaluation in vr display. In *SID Symposium Digest of Technical Papers*, volume 48, pages 1154–1156. Wiley Online Library, 2017.

[3] Antonio Criminisi, Patrick Pérez, and Kentaro Toyama. Region filling and object removal by exemplar-based image inpainting. *IEEE Trans. Image Processing*, 13(9):1200–1212, 2004.

[4] Ismael Daribo and Béatrice Pesquet-Popescu. Depth-aided image inpainting for novel view synthesis. In *IEEE Inte workshop on multimedia signal processing*, pages 167–170, 2010.

[5] Gyorgy Denes, Kuba Maruszczyk, George Ash, and Rafał K Mantiuk. Temporal resolution multiplexing: Exploiting the limitations of spatio-temporal vision for more efficient vr rendering. *IEEE Trans. Vis. and Comp. Graph*, 25(5):2072–82, 2019.

[6] Piotr Didyk, Tobias Ritschel, Elmar Eisemann, Karol Myszkowski, and Hans-Peter Seidel. Adaptive image-space stereo view synthesis. In *VMV*, number 1, 2, pages 299–306, 2010.

[7] Piotr Didyk, Tobias Ritschel, Elmar Eisemann, Karol Myszkowski, Hans-Peter Seidel, and Wojciech Matusik. A luminance-contrast-aware disparity model and applications. *ACM Trans Graph (Proc. SIGGRAPH Asia)*, 31(6), 2012.

[8] William Donnelly and Andrew Lauritzen. Variance shadow maps. In *Proc. ACM I3D*, pages 161–165, 2006.

[9] Alexei A Efros and Thomas K Leung. Texture synthesis by non-parametric sampling. In *ICCV*, volume 2, pages 1033–1038, 1999.

[10] Daniel Evangelakos and Michael Mara. Extended timewarp latency compensation for virtual reality. In *I3D*, page 193–94, 2016.

[11] Jeremy Freeman and Eero P Simoncelli. Metamers of the ventral stream. *Nature Neuroscience*, 14(9):1195–1201, 2011.

[12] William T Freeman, Edward H Adelson, et al. The design and use of steerable filters. *IEEE PAMI*, 13(9):891–906, 1991.

[13] Leon A Gatys, Alexander S Ecker, and Matthias Bethge. Image style transfer using convolutional neural networks. In *CVPR*, pages 2414–2423, 2016.

[14] Josselin Gautier, Olivier Le Meur, and Christine Guillemot. Depth-based image completion for view synthesis. In *EDTV*, pages 1–4, 2011.

[15] Steven J Gortler, Radek Grzeszczuk, Richard Szeliski, and Michael F Cohen. The lumigraph. In *SIGGRAPH*, pages 43–54, 1996.

[16] John A. Greenwood, Peter J. Bex, and Steven C. Dakin. Positional averaging explains crowding with letter-like stimuli. *Proc NAS US*, 106(31):13130–5, 2009.

[17] David J Heeger and James R Bergen. Pyramid-based texture analysis/synthesis. In *Proc. SIGGRAPH*, pages 229–238, 1995.

[18] Jan Herling and Wolfgang Broll. High-quality real-time video inpaintingwith pixmix. *IEEE Transactions on Visualization and Computer Graphics*, 20(6):866–879, 2014.

Fig. 9: Temporal up-sampling in the periphery on a foveated displays. The top row shows a Varjo XR-3-like setup: a dense fovea (ca. 100 pixels per degree) at high refresh rate (90 Hz) and a sparse periphery (10 ppd) at low refresh (30 Hz), up-sampled in time. the second row is our method, to be compared to the reference in the third row, and push-pull in the last row. When fixating the yellow dot on a A4 printout in a stretched arm's distance, blur from push-pull is perceived in the periphery, while ours appears plausible.

Fig. 10: Limitation of our method: although it performs well on textured regions (left, center), sharp oriented edges are not synthesised correctly in the disoccluded region (right).

[19] Youichi Horry, Ken-Ichi Anjyo, and Kiyoshi Arai. Tour into the picture: using a spidery mesh interface to make animation from a single image. In *SIGGRAPH*, pages 225–32, 1997.

[20] Xun Huang and Serge Belongie. Arbitrary style transfer in real-time with adaptive instance normalization. In *ICCV*, pages 1501–1510, 2017.

[21] Justin Johnson, Alexandre Alahi, and Li Fei-Fei. Perceptual losses for real-time style transfer and super-resolution. In *ECCV*, pages 694–711, 2016.

[22] Hans Knutsson and C-F Westin. Normalized and differential convolution. In *CVPR*, pages 515–523, 1993.

[23] Johannes Kopf, Kevin Matzen, Suhib Alsisan, Ocean Quigley, Francis Ge, Yangming Chong, Josh Patterson, Jan-Michael Frahm, Shu Wu, Matthew Yu, et al. One shot 3D photography. *ACM Trans. Graph. (Proc. SIGGRAPH)*, 39(4), 2020.

[24] Ares Lagae, Peter Vangorp, Toon Lenaerts, and Philip Dutré. Procedural isotropic stochastic textures by example. *Computers & Graphics*, 34(4):312–321, 2010.

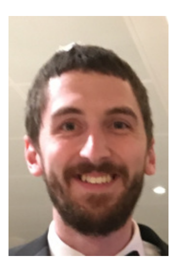
[25] Lin Liang, Ce Liu, Ying-Qing Xu, Baining Guo, and Heung-Yeung Shum. Real-time texture synthesis by patch-based sampling. *ACM Trans Graph*, 20(3):127–150, 2001.

[26] Timothy Lottes. FXAA. White Paper, NVIDIA, 2009.

[27] Rafał K Mantiuk, Gyorgy Denes, Alexandre Chapiro, Anton Kaplanyan, Gizem Rufo, Romain Bachy, Trisha Lian, and Anjul Patney. Fovvideovdp: A visible difference predictor

for wide field-of-view video. *ACM Transactions on Graphics (TOG)*, 40(4):1–19, 2021.
[28] William R Mark, Leonard McMillan, and Gary Bishop. Post-rendering 3d warping. In *I3D*, pages 7–ff, 1997.
[29] Ricardo Marroquim, Martin Kraus, and Paulo Roma Cavalcanti. Efficient point-based rendering using image reconstruction. In *SPBG*, pages 101–108, 2007.
[30] Shohei Mori, Okan Erat, Wolfgang Broll, Hideo Saito, Dieter Schmalstieg, and Denis Kalkofen. Inpaintfusion: incremental rgb-d inpainting for 3d scenes. *IEEE Transactions on Visualization and Computer Graphics*, 26(10):2994–3007, 2020.
[31] Joerg H Mueller, Philip Voglreiter, Mark Dokter, Thomas Neff, Mina Makar, Markus Steinberger, and Dieter Schmalstieg. Shading atlas streaming. *ACM Trans. Graph.*, 37(6):1–16, 2018.
[32] Diego Nehab, Pedro V Sander, Jason Lawrence, Natalya Tatarchuk, and John R Isidoro. Accelerating real-time shading with reverse reprojection caching. In *Graphics Hardware*, volume 41, pages 61–62, 2007.
[33] Makoto Okabe, Keita Noda, Yoshinori Dobashi, and Ken Anjyo. Interactive video completion. *IEEE Computer Graphics and Applications*, 40(1):127–139, 2019.
[34] Ken Perlin. An image synthesizer. *ACM Siggraph Computer Graphics*, 19(3):287–296, 1985.
[35] Javier Portilla and Eero P Simoncelli. A parametric texture model based on joint statistics of complex wavelet coefficients. *Int J Comp Vis*, 40(1):49–70, 2000.
[36] Charles Poynton. *Digital Video and HD: Algorithms and Interfaces*. Morgan Kaufmann, 2012.
[37] Bernhard Reinert, Johannes Kopf, Tobias Ritschel, Eduardo Cuervo, David Chu, and Hans-Peter Seidel. Proxy-guided image-based rendering for mobile devices. In *Comp. Graph. Forum*, volume 35, pages 353–362, 2016.
[38] Laura Walker Renninger and Jitendra Malik. When is scene identification just texture recognition? *Vision research*, 44 (19):2301–2311, 2004.
[39] R Rosenholtz. Capabilities and limitations of peripheral vision. *Annual review of vision science*, 2:437, 2016.
[40] Ruth Rosenholtz, Jie Huang, Alvin Raj, Benjamin J. Balas, and Livia Ilie. A summary statistic representation in peripheral vision explains visual search. *J Vision*, 12(4):14–14, apr 2012.
[41] Anita M Schmid, Keith P Purpura, Ifije E Ohiorhenuan, Ferenc Mechler, and Jonathan D Victor. Subpopulations of neurons in visual area v2 perform differentiation and integration operations in space and time. *Frontiers in systems neuroscience*, 3:15, 2009.
[42] Thomas Schöps, Martin R Oswald, Pablo Speciale, Shuoran Yang, and Marc Pollefeys. Real-time view correction for mobile devices. *IEEE transactions on visualization and computer graphics*, 23(11):2455–2462, 2017.
[43] Jonathan Shade, Steven Gortler, Li-wei He, and Richard Szeliski. Layered depth images. In *SIGGRAPH*, pages 231–242, 1998.
[44] Meng-Li Shih, Shih-Yang Su, Johannes Kopf, and Jia-Bin Huang. 3d photography using context-aware layered depth inpainting. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.
[45] Meng-Li Shih, Shih-Yang Su, Johannes Kopf, and Jia-Bin Huang. 3D photography using context-aware layered depth inpainting. In *CVPR*, pages 8028–38, 2020.
[46] Hans Strasburger, Ingo Rentschler, and Martin Jüttner. Peripheral vision and pattern recognition: A review. *J Vision*, 11 (5):13–13, 2011.
[47] Zinovi Tauber, Ze-Nian Li, and Mark S Drew. Review and preview: Disocclusion by inpainting for image-based rendering. *IEEE Trans Systems, Man, and Cybernetics C*, 37 (4):527–540, 2007.
[48] David R. Walton, Rafael Kuffner Dos Anjos, Sebastian Friston, David Swapp, Kaan Akşit, Anthony Steed, and Tobias Ritschel. Beyond blur: Ventral metamers for foveated rendering. *ACM Trans. Graph. (Proc. SIGGRAPH 2021)*, 40 (4), 2021.
[49] Li-Yi Wei, Sylvain Lefebvre, Vivek Kwatra, and Greg Turk. State of the art in example-based texture synthesis. In *Eurographics STAR*, pages 93–117, 2009.
[50] Lei Yang, Yu-Chiu Tse, Pedro V. Sander, Jason Lawrence, Diego Nehab, Hugues Hoppe, and Clara L. Wilkins. Image-based bidirectional scene reprojection. *ACM Trans. Graph.*, 30(6):1, 2011.
[51] Zili Yi, Qiang Tang, Shekoofeh Azizi, Daesik Jang, and Zhan Xu. Contextual residual aggregation for ultra high-resolution image inpainting. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7508–7517, 2020.
[52] Jiahui Yu, Zhe Lin, Jimei Yang, Xiaohui Shen, Xin Lu, and Thomas S Huang. Free-form image inpainting with gated convolution. In *CVPR*, pages 4471–4480, 2019.
[53] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *CVPR*, pages 586–95, 2018.
[54] Tinghui Zhou, Richard Tucker, John Flynn, Graham Fyffe, and Noah Snavely. Stereo magnification: learning view synthesis using multiplane images. *ACM Trams Graph. (Proc. SIGGRAPH)*, 37(4):1–12, 2018.

Rafael Kuffner dos Anjos is a Lecturer in Computer Graphics at University of Leeds. He obtained his PhD (2018) from the University of Lisbon where he focused on Video-based rendering for point cloud data, applied to VR/AR. He has previously worked at Victoria University of Wellington on computer graphics techniques for entertainment industry pipelines, and at University College London, focusing on perception and real-time rendering for VR/AR. His current research interests are Image based rendering, Perception, Point-clouds, VR/AR.



David R. Walton was awarded his EngD from University College London in 2019, focusing on applying computer vision and sensing hardware to improve AR rendering. From 2020 he has worked at University College London as a Research Fellow. His research interests include foveated rendering, human visual perception, novel display technologies and virtual & augmented reality.

Kaan Akşit is an Associate Professor in the computer science department at University College London. Kaan received his PhD in electrical engineering at Koç University, Turkey, in 2014. Kaan researches the intersection of light and computation, including computational approaches in imaging, graphics, fabrication and displays. Kaan's research works are most known among the optics and graphic community for his contributions to display technologies dedicated to virtual reality, augmented reality, and three-dimensional displays with glasses and without glasses. He worked as a research intern in Philips Research, the Netherlands, and Disney Research, Switzerland, in 2009 and 2013, respectively. He was a scientist at NVIDIA, between 2014 and 2020. He is the recipient of Emerging Technologies best in show awards in SIGGRAPH 2019 and SIGGRAPH 2018, DCEXPO special prize in SIGGRAPH 2017, and among the best papers in IEEE VR 2021, IEEE VR 2019, ISMAR 2018, and IEEE VR 2017.

Sebastian Friston received his EngD from University College London in 2017. He is a currently a Research Associate in the Virtual Environments and Computer Graphics group at University College London. His work has received the IEEE VR Best Dissertation Award (2018). His research interests are in how to build high fidelity virtual worlds.

David Swapp received his PhD in 1996 from the University of Aberdeen. He is currently a Senior Research Fellow at University College London where he collaborates on a wide range of research and experimental work, with particular interest in how knowledge of human perceptual mechanisms can be exploited to overcome technical limitations of VR systems.

Anthony Steed Professor Anthony Steed, Member of IEEE,was awarded his PhD in 1996 from Queen Mary College, University of London in the area of immersive virtual reality systems. From 1996 he has been at University College London, first as a Research Fellow, then as Lecturer, Senior Lecturer, Reader and, since 2009, Professor. His research interests range from real-time computer graphics systems, through novel displays, to user-evaluation techniques. In 2018-2019 he was a Visiting Scientist at Microsoft Research, Redmond. In 2019 he visited Christchurch, New Zealand, supported by an Erskine Fellowship. He was the recipient of the IEEE VGTC's 2016 Virtual Reality Technical Achievement Award.

Tobias Ritschel received his PhD from Saarland University (MPI) in 2009. He was a post-doctoral researcher at Telecom ParisTech / CNRS 2009-10 and a Senior Researcher at MPI 2010-15. Tobias was appointed Senior Lecturer at University College London in 2015 where he was named Full Professor of Computer Graphics in 2019. His work has received the EG Dissertation (2010) and Young Researcher Award (2014). His interests include Image Synthesis and Human Visual Perception, now frequently including applied AI.