InpaintFusion: Incremental RGB-D Inpainting for 3D Scenes

Shohei Mori, Okan Erat, Wolfgang Broll, *Member, IEEE,*
Hideo Saito, Dieter Schmalstieg, *Senior Member, IEEE,* and Denis Kalkofen, *Member, IEEE*

Abstract—State-of-the-art methods for diminished reality propagate pixel information from a keyframe to subsequent frames for real-time inpainting. However, these approaches produce artifacts, if the scene geometry is not sufficiently planar. In this paper, we present InpaintFusion, a new real-time method that extends inpainting to non-planar scenes by considering both color and depth information in the inpainting process. We use an RGB-D sensor for simultaneous localization and mapping, in order to both track the camera and obtain a surfel map in addition to RGB images. We use the RGB-D information in a cost function for both the color and the geometric appearance to derive a global optimization for simultaneous inpainting of color and depth. The inpainted depth is merged in a global map by depth fusion. For the final rendering, we project the map model into image space, where we can use it for effects such as relighting and stereo rendering of otherwise hidden structures. We demonstrate the capabilities of our method by comparing it to inpainting results with methods using planar geometric proxies.

Index Terms—Diminished Reality, Inpainting, Fusion, SLAM.

1 Introduction

Diminished reality (DR) allows removing objects from the user's perspective view and uncovering otherwise hidden structure in the user's physical environment [1], [2]. Occluded pixels in the region of interest (ROI) are restored from either multi-view observations or by inpainting. Multi-view approaches directly observe the background at different points of view, either in preprocessing [3], [4] or online, using additional cameras [5], [6]. The resulting reconstructions represent the real situation accurately, providing a high level of confidence and high quality rendering [5], [6], [7].

However, multi-view DR cannot restore unobserved areas. Inpainting can overcome this problem. It uses pixels in the vicinity of the ROI and, therefore, does not require additional cameras or pre-recorded observations. If "hallucinated" pixels are acceptable, inpainting has considerable benefits over observation-based methods, in particular, for mobile purposes, where offline preparation is not feasible.

Inpainting is easily performed in image space or in planar neighborhoods in object space, but this can limit the supported application cases. For example, applications that require temporal or spatial coherence between frames, such as the rendering of stereoscopic images or relighting of the background, are not possible without information about the underlying 3D structure. For this reason, some inpainting systems assume a 3D space, for example, by estimating a dominant plane and performing inpainting operations on a planar embedding. If the dominant plane can be tracked throughout a sequence of frames, the inpainted images can be projected back into the user's perspective view. Such an approach is sufficient for providing plausibly DR, but only if the scene is flat and occlusions can be safely ignored. Even if the inpainted result is deformed [8], such a deformation changes only the appearance and never fits the geometry.

In this paper, we present a novel approach for augmented reality (AR), *InpaintFusion*, which inpaints both color and depth information in occluded regions. Given an RGB-D frame, our method simultaneously searches in the color and depth-gradient channel. We minimize a cost function consisting of a color term and a spatial term in image space [9], but extend it with a geometric term in 3D, which analyzes depth gradients to ensure global consistency of the inpainted geometry. Spatio-temporal consistency is ensured by reprojecting inpainted frames into subsequent frames and merging them using volumetric fusion.

InpaintFusion enables to change the visualization of the physical environment in addition to adding virtual objects. In Fig. 1, we removed the physical horse from the scene by inpainting its color and depth values. Instead of the horse, a car with head lights is inserted in the scene to demonstrate the ability to relight the inpainted color and depth information. Our work makes the following contributions.

- We introduce a novel 3D geometric term based on depth gradient sampling, which enables consistent color and geometry inpainting in arbitrary 3D scenes.
- We present a novel system for combining fusion from SLAM with multi-keyframe inpainting. Our approach synthesizes a globally consistent surfel map from depth inpainting that is applied to keyframes.
- We demonstrate how a globally consistent geometric model enables AR rendering effects, e.g., relighting.
- We provide the results of a user evaluation showing the superiority of the proposed approach compared to approaches that use planar geometric proxies.

- *S. Mori, O. Erat, D. Schmalstieg, and D. Kalkofen are with the Institute of Computer Vision and Graphics, Graz University of Technology, Austria. E-mail: see http://www.michaelshell.org/contact.html*
- *W. Broll is with the Virtual Worlds and Digital Games Group, Ilmenau University of Technology, Germany.*
- *H. Saito is with the Department of Information and Computer Science, Keio University, Japan.*

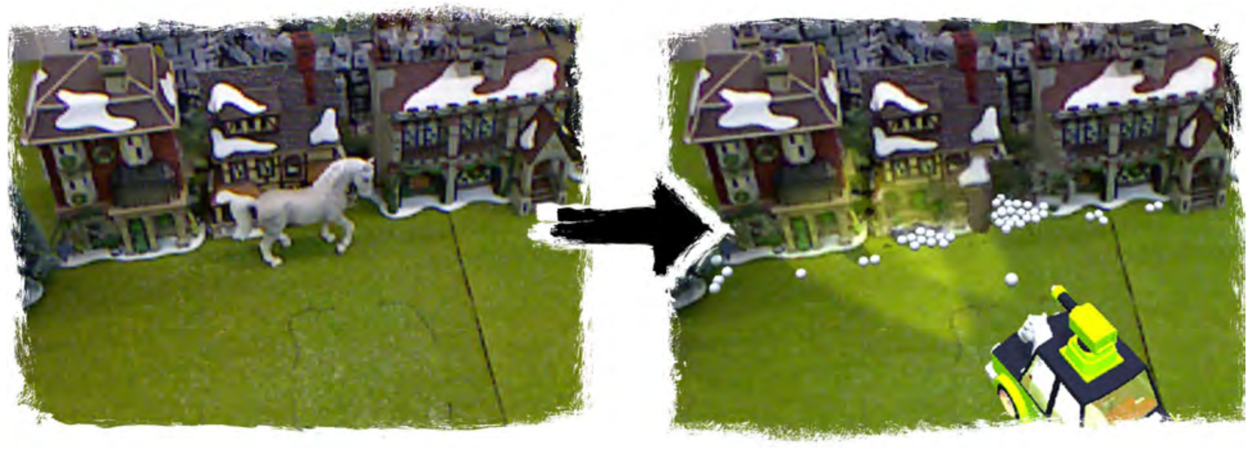


Fig. 1. 3D diminished reality followed by augmented reality rendering. Our system, *InpaintFusion*, is able to remove objects and inpaint color and depth information. While previous attempts for real-time inpainting are limited to color channel inpainting only, *InpaintFusion* supports arbitrary depth channel inpainting as well. As it provides depth information, 3D augmented reality rendering in the inpainted environment becomes possible. This example shows a real white horse in front of a set of houses (left). We inpaint the color and depth values of the horse (right). This enables us to add a virtual car with head lights to illuminate the inpainted regions, together with white balls which can interact with the inpainted region by bouncing of the walls.

2 Related work

The simplest form of DR, using only a single image, replaces pixels in a target region T, $T \in I$, of an image I with pixels from sources $S \in I$. Therefore, we need to find the transformation $f : T \rightarrow S$ that preserves consistency in the appearance between the target region T and the remaining image $\overline{T} = I \setminus T$. Furthermore, DR methods need to support motion in 3D space with six degrees of freedom (6DOF). This implies that, after inpainting, T and $\overline{T}$ need to be consistent under arbitrary motion of the camera. Previous DR methods mainly differ in how the function f is defined and which sources S are considered. Therefore, in this section, we review previous approaches (see TABLE 1).

2.1 Multi-view approaches

One direction of research has focused on rendering occluded pixels from multiple different live video observations of the hidden area, while another direction first captures the scene from multiple camera positions using a single camera. An early example of the former case is the multi-view paraperspective projection model proposed by Zokai et al. [6] that uses additional calibrated cameras as S to search for background patches in T with a similar appearance. Meerits and Saito [15] use additional RGB-D frames from a Microsoft Kinect sensor as S to observe the background with depth information. The work of Cosco et al. [3] creates DR from multiple images that have been captured over time. They propose a system recording images as S, before the object to be diminished is placed in the scene. While Cosco et al. use the multi-view data immediately after capturing, Li et al. [4] use older images from Internet photo collections, registered in 3D space as S.

The above methods assume calibrated multi-view cameras to define the mapping f under epipolar constraint [16]. This enables a fast pixel search in S at the price of relying on dense observations, which may have to be generated in advance or at runtime using additional cameras. Either restriction makes these approaches difficult to apply to mobile applications.

2.2 Video inpainting

A more flexible approach for DR is inpainting, which can be defined as the global optimization of the transformation function $f : T \rightarrow S$ in which $S \equiv \overline{T}$, i.e., $f : T \rightarrow \overline{T}$ [9].

Inpainting for DR originates from research on video restoration. The primary difference between image and video inpainting is that video inpainting makes use of the pre-recorded image sequence as an inpainting source S [17], instead of just a single frame. Thus, it can be defined as a global optimization problem of finding the best transformation function $f : T \rightarrow S$ where $S \equiv \overline{T}_i$ at frame $F(i)$.

Wang et al. [18] presented pioneering work in this area. They separate the pre-recorded scene into several layers using dominant optical flow, and showed that rendering all layers except one results in a scene without the selected object. Lepetit et al. [19] take pixels from $\overline{T}_j$ in frame $F(j), i \neq j$, to inpaint T_i by reprojection via a reconstructed background triangle mesh. Shen et al. [20] find a linearly moving foreground object in a geometrically aligned temporal texture space and propagate non-occluded pixels in $\overline{T}_j$ to the occluded pixels in T_i. Klose et al. [21] use a point cloud for inpainting, where point reconstructions from $\overline{T}_j$ are sampled through pre-defined filters to fill in T_i.

Although video inpainting methods generate plausible results, they cannot be used in DR applications, since they rely on costly global optimizations and consider both past and future frames. DR must be able to react instantaneously to changes in the user's viewpoint, using only past information, while maintaining coherent visual appearance over time. Furthermore, usage of past frames is typically limited to the previous frame or a small number of frames.

2.3 Image and depth inpainting

In contrast to video inpainting, image inpainting takes pixel information only from a single image. The most popular image inpainting approaches are based on the idea of searching

TABLE 1
Qualitative comparison of literature on inpainting for DR.

Literature	Scene geometry	Object detection	Object tracking	Depth for AR
Siltanen [10]	Plane	Marker region	6DoF marker tracking	No
Korkalo et al. [11]	Plane	Marker region	6DoF marker tracking	No
Herling and Broll [9], [12]	Plane	Interactive drawing (one-view)	2D contour tracking	No
Kawai et al. [13]	Plane(s)	User drawing (multi-views)	6DoF SLAM	No
Siltanen [14]	Plane(s)	User drawing (one-view)	6DoF SLAM	No
Kawai et al. [8]	Curved plane	Marker region	6DoF marker tracking	No
Proposed method*	3D scene	Interactive 3D labeling	6DoF SLAM	Yes

*Only our proposed method uses RGB-D frame inputs for dense 3D reconstruction.

for patterns in the image which are similar to a region placed over the boundary of T and $\overline{T}$ [22]. The creators of the PatchMatch method [23], [24] report on two key insights for finding a near optimal f: They use randomized searching for corresponding patches in $\overline{T}$, and they make use of propagation of the searched offsets to the adjacent pixels in T. These two insights enable generating consistent reconstructions.

However, PatchMatch is not designed for predictable real-time performance, and does not consider temporal coherence over image sequences. Therefore, Herling and Broll [12] proposed PixMix, a method relying on frame-to-frame propagation of patches to accelerate the search and ensure temporal coherence. Later, they improved image quality and runtime of their method [9] by applying a homography transformation (estimated between an earlier keyframe and the current frame) to the reference map $f : T \rightarrow \overline{T}'$, where $\overline{T}'$ represents T transformed by the homography. Kunert et al. [25] extended the method by combining it with observed background pixels. Kawai et al. [13] and Siltanen [14] also extended this strategy to enable processing of several planes in parallel. Kawai et al. [8] furthermore proposed an inpainting algorithm that deforms inpainted results using feature point tracking.

All these attempts assume that the scene is locally planar. While this notion makes it easier to obtain real-time performance, it cannot recover depth information in T. Advanced AR rendering typically requires the evaluation of lighting, occlusion and other view-dependent phenomena [26], [27], as well as image synthesis for stereoscopic displays [28]. Without restoring proper depth information in the inpainted area, such rendering methods cannot be properly supported.

Our work is also conceptually related to depth densification. Unlike offline structure-from-motion methods, AR requires densification to operate in real-time. State-of-the-art methods densify sparse SLAM maps [29] or perform real-time short-baseline stereo matching [30]. We could use such methods as alternative forms of reconstruction, but, of course, they cannot deal with unobserved areas.

Recently, convolutional neural networks (CNN), in particular, generative adversarial networks [31], have shown great promise for complex image synthesis [32], [33], [34], [35]. Inpainting based on CNN essentially uses a database of trained feature as S. Such approaches have also been shown to be able to generate depth maps from color images [36], [37] or inpaint RGB-D images [38]. However, CNN typically requires images to be resized before feeding them into the network, and, again, on the output side. These implicit resampling steps make the results prone to aliasing problems, when the inpainted area changes with perspective distortion (another resampling step), as the camera pose changes from frame to frame. In contrast, our approach inpaints color and depth using a conventional patch representation, which does not have to be scaled or resampled. It also has the advantage that it works instantaneously without requiring extensive training databases to be collected and processed.

3 Method

In this section, we give an overview of our method, beginning with a concise problem statement. Please note that we rely on the notation introduced in section 2.

3.1 The problem of depth inpainting

Previous inpainting methods rely on homography warping and thus assume planar scenes. Once a keyframe $F(0)$ is selected, the system inpaints the ROI as specified by the user. Given a relationship between $F(0)$ and the current frame $F(i)$ by a homography, $F(0)$ is transformed into the current frame as $F'(0)$, and the system overlays $F'(0)$ onto $F(i)$. Since all the pixels in T_0 of $F(0)$ are potentially visible in $F'(0)$, the system can refine $F'(0)$ using new pixel samples in $F'(i)$ with high constraints on the patch appearance, in order not to break the texture [9], or progress with pixel searching in $F(0)$ [13]. In other words, assuming the pixels' spatial relationship will be preserved by a homography transformation, these approaches can improve the inpainting over time regardless of viewpoint changes.

For 3D inpainting, both update rules do not work. The 3D structure of the scene induces occlusions between the projected pixels of $F(0)$ to $F(i)$, causing new missing pixels to appear in $F(i)$. These additional missing pixels needs to be inpainted in the projected keyframe $F'(0)$.

Refining such hallucinated pixels does not work, since they have been collected from various sources, which are not necessarily consistent beyond the originally copied pixels. For this reason, previous work has resorted to manual labeling for additional constraints [9] or indirectly infering additional structure by decomposing the scene into multiple planes [13]. Refining $F(0)$ does not resolve the problem, either, since the newly found missing pixels are mostly invisible at $F(0)$ due to occlusion. Fig. 2 depicts this problem.

Consequently, we need a novel approach for 3D inpainting, which incrementally fills in background 3D structure without destroying previously inpainted color and structure. To this end, we propose to combine fusion from SLAM with multi-keyframe inpainting. A novel fusion method merges structural information of all inpainted keyframes

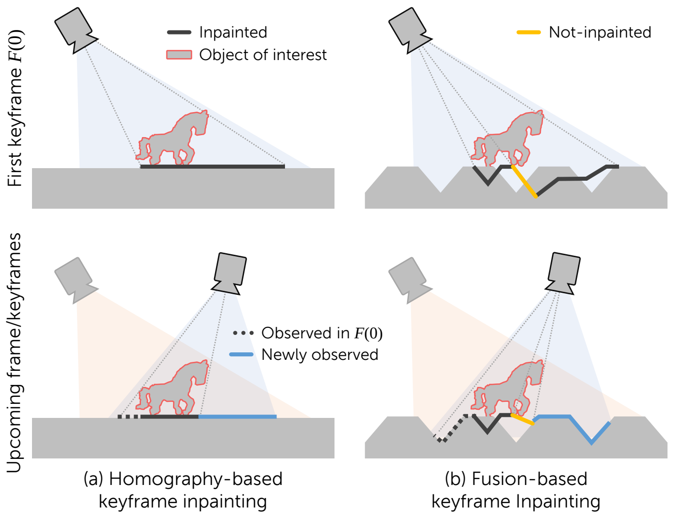


Fig. 2. Comparison between homography-based keyframe inpainting and our multi-keyframe inpainting. (a) Assuming all pixels in a keyframe are visible, a homography to a single keyframe may be sufficient. This approach can even further update the keyframe over time by warping it. (b) Assuming a non-planar background, we fuse multiple keyframes. If the spatial relationship of the pixels changes due to occlusion, it is difficult to refine the pixels using the projected keyframe. We cannot project the current frame back to one of the keyframes, either, since currently visible pixels may not be visible in the keyframe anymore.

into one consistent global map. We also present a rendering scheme to synthesize multiple inpainted color frames relying on labels from the SLAM system to minimize inpainted regions and to use observed background regions instead, if available.

3.2 System overview

Our system supports per-pixel recovery of color and depth information in the unobservable region T in real time. For a frame F, it performs exemplar-based inpainting of a region T by copying information from $\overline{T}$, in both color and geometry domains, on top of a SLAM system [39]. Fig. 3 illustrates our system architecture. Our system pipeline has the following stages:

Scene scanning using SLAM. We rely on a SLAM system to obtain an RGB-D frame $F(i) = (\mathcal{C}_i, \mathcal{D}_i, \mathcal{V}_i, \mathcal{N}_i, \mathbf{M}_i)$, consisting of a color buffer $\mathcal{C}_i$, a depth buffer $\mathcal{D}_i$, a vertex buffer $\mathcal{V}_i$, a normal buffer $\mathcal{N}_i$, and a 6DOF camera pose $\mathbf{M}_i$, expressed as an $\mathbb{SE}3$ transformation matrix. The frame $F(i)$ is fused over time into a global map $\mathcal{G}$ represented as a collection of surfels [40], i.e., the measurement $F(i)$ updates $\mathcal{G}$ in accordance with the previous work [39].

Object labeling. While SLAM runs, the user interactively labels a ROI in 2D screen space, which will be preserved in $\mathcal{G}$ and generate 2D target regions when projected.

Keyframe insertion. For stable inpainting over frames, we utilize keyframes. Our goal is to fill in all missing pixels in every novel viewpoint. Therefore, our system inserts a new keyframe when the pose diverges too much from the closest keyframe, while pixels remain missing in the ROI. Since the inpainting process is too costly to be completed before the next frame arrives, inpainting is performed in an asynchronous background thread.

Keyframe propagation. Here, a mapping of the transformation function f_{k-m} from the closest keyframe $F(k-m)$ to a newly selected keyframe $F(k)$ is derived to initialize the transformation function f_k. For the first keyframe $F(0)$, the transformation function f_0 is initialized with random values.

Keyframe inpainting. An inpainted keyframe $\hat{F}(k) = (\hat{\mathcal{C}}_k, \hat{\mathcal{D}}_k, \hat{\mathcal{V}}_k, \hat{\mathcal{N}}_k, \mathbf{M}_k)$ is computed for the new keyframe $F(k)$ by minimizing a cost function over all pixels $\mathbf{u} \in T_k$ with the given f_k, or, otherwise, from a random guess.

Inpainted keyframe fusion. The inpainted keyframe is passed to the SLAM system and fused with existing surfels. To avoid interference with tracking, inpainted keyframes are fused only in the ROI.

View-dependent keyframe blending. Labeled surfels are projected to the current frame $F(i)$ at $\mathbf{M}_i$ to obtain the ROI T_i, which is filled with pixels from multiple inpainted keyframes. The keyframes are projected to $F(i)$ via the fused inpainted surfels and blended depending on the viewpoint.

AR rendering. Our system can provide the inpainted RGB-D frame or the inpainted global world model for additional AR rendering. In the following sections, we describe each of these stages in detail.

3.3 Scene scanning using SLAM

We use the dense map and device pose provided by the SLAM system of Keller et al. [39] to analyze the scene color and geometry, but extract some additional data from the global map $\mathcal{G}$. Each point in our global map $\mathcal{G}$ is represented as surfel $\mathcal{S}$ which contains a 24-bit RGB color $\mathbf{c} = [R, G, B]^{\mathrm{T}}$, a 3D position $\mathbf{p} = [X, Y, Z]^{\mathrm{T}}$, a normal $\mathbf{n} = [n_x, n_y, n_z]^{\mathrm{T}}$, a radius r, a depth confidence value $conf$, an index to distinguish a surfel among the rest of the surfels, and a label $l \in \{L_O, L_{IP}, L_{ROI}\}$, which classifies scene points as observed (L_O), inpainted (L_{IP}), or belonging to the ROI (L_{ROI}).

We smooth the sensor depth map $\mathcal{D}_i$ using a bilateral filter [41] and derive a vertex map $\mathcal{V}_i$ and a normal map $\mathcal{N}_i$. Subsequently, we estimate the pose $\mathbf{M}_i$ using an iterative closest point algorithm [42], aligning $\mathcal{D}_i$ with a virtual depth map, which we generate by reprojecting the global scene map $\mathcal{G}$ fused over time, into frame $F(i-1)$. The inpainted region is not tracked; only surfels with L_O or L_{ROI} are used for generating the virtual maps. The vertex map $\mathcal{V}_i$ is derived from the smoothed depth map, and the normal map $\mathcal{N}_i$ is fused into the global map $\mathcal{G}$ using weights derived from the observed timing and the estimated confidence [39]. Initially, all the surfels are assigned the value L_O, until the user categorizes a surfel as L_{ROI}.

3.4 Object labeling

One application of DR is removing undesirable objects, such as markers [8], [10], [11], pedestrians [4], cars [7], or any other category of objects that can be pre-trained [43]. Another type of application lets the user specify the ROI, e.g., by painting coarse strokes [9], and then segmenting

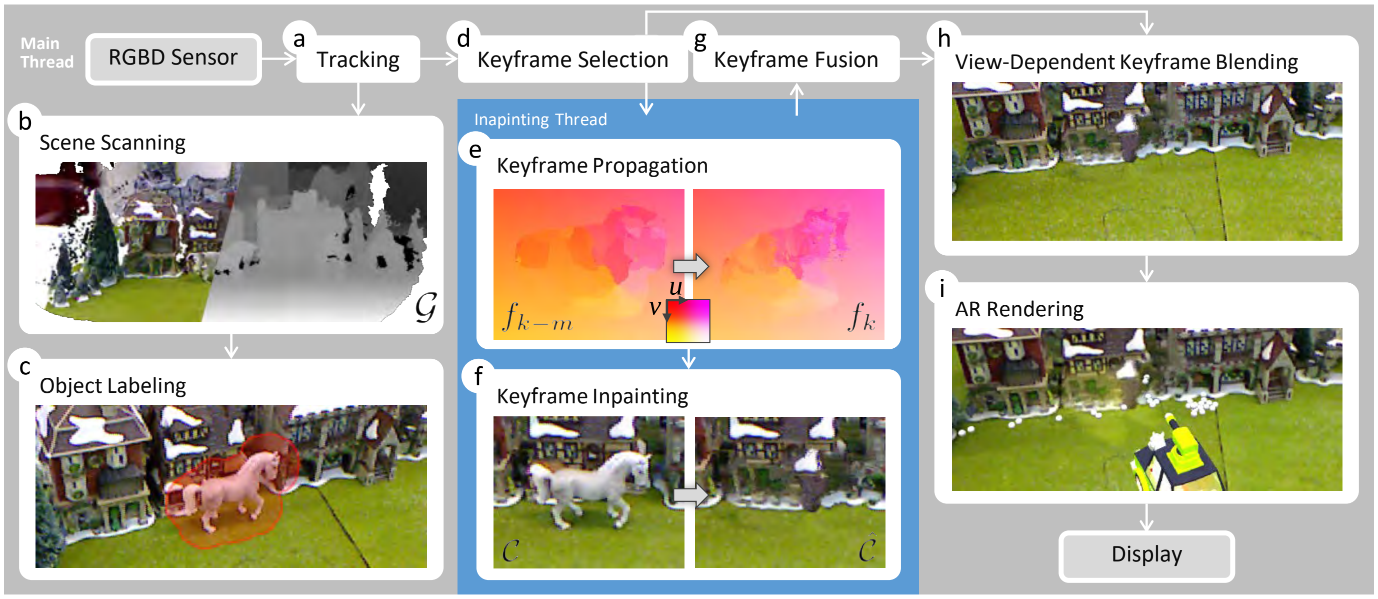


Fig. 3. Overview of our *InpaintFusion* framework. Our method operates on an RGB-D frame. We first align the input frame in the world coordinate system by estimating the camera pose (a). This enables us to map pixels from the input frame to the 3D model and to a keyframe (b). Once the user labels the object of interest on a screen (c), the system preserves a keyframe (d) and inpaints the keyframe. We use previous inpainting results in keyframes to coherently map pixels over time (e). Subsequently, we search for an optimal set of pixel values which fill in the remaining unknown RGB-D values (f). To generate consistent 3D information, we fuse the inpainted depth map into a surfel map $\mathcal{G}$ (g). Finally, we blend available keyframes on the inpainted surfaces (h) and apply rendering effects to the 3D model (i).

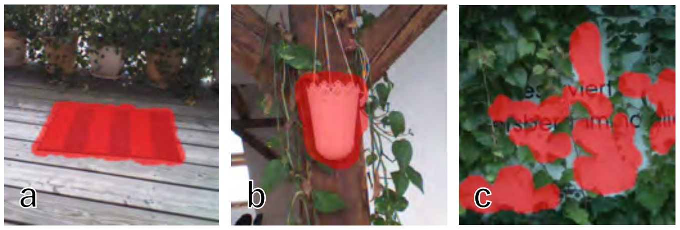


Fig. 4. Labeling the 3D object of interest with 2D user interaction. The system translates user's pointing into a 2D label map, and fuses the label map into the global surfel map $\mathcal{G}$. Projecting such labeled surfel map to the tracked frame results in 2D registered label map for (a) a plane, or (b) a more complex object. (c) Our system allows users to label multiple objects.

and tracking these objects using image gradients. Gradient-based segmentation is fast and tends to work well on a planar background and a planar object of interest. In more complex environments, such as the multi-plane inpainting of Kawai et al. [13], interaction gets more complicated – the user must not only encircle the object, but also trim the segmentation when the view has changed too much, since the 3D object shape cannot be determined with sufficient accuracy from a sparse map from a visual SLAM system.

Since we have access to a dense map, our method can directly label the map by projecting the user's 2D input onto the depth map. Fig. 4 shows an example of our labeling result. Inspired by incremental 3D segmentation [44], our system first encodes label information in pixels and then fuses the 2D label map into the global map. For this purpose, the user coarsely traces the object on a 2D screen to provide the input $\mathbf{u}_{user}$. Our system defines a 3D bounding disc with a center $\mathcal{V}(\mathbf{u}_{user})$, a radius r_{ROI}, and a thickness d_{ROI} along a surfel normal. Pixels that have vertices within the disc are labeled as L_{ROI}, and the rest, as L_O.

$$\mathcal{L}(\mathbf{u}) = \begin{cases} L_{ROI}, & \text{if } ||\mathbf{u}_{user} - \mathbf{u}||_2 < r_{ROI} \wedge \\ & (\mathcal{V}(\mathbf{u}_{user}) - \mathcal{V}(\mathbf{u})) \cdot \mathcal{N}(\mathbf{u}) < d_{ROI} \\ L_O, & \text{otherwise} \end{cases} \quad (1)$$

The 2D label map $\mathcal{L}$ is fused with the 3D global map $\mathcal{G}$. To give safe margins for the ROI, the system dilates the region where such surfels are projected to the screen. The user can set a value for r_{ROI} that corresponds to the effective range of the labeling on the screen. For d_{ROI}, one may set a sensor depth uncertainty [39] that describes in which range the specified depth should cover surfels in the global map.

3.5 Keyframe insertion

DR requires inpainting to be temporally coherent, which is usually addressed by using keyframes [9], [13], [14]. One can inpaint a frame as a keyframe and preserve it for future frames by warping the inpainted frame to the current frame. In the case of planar scenes, homography warping is sufficient as a geometric representation, as long as the ROI is tracked [13], [14] and pixel colors are referenced from visible regions within the frame [9]. In other words, pixel searching by inpainting need not to be repeated after inpainting the initial keyframe, and this strategy significantly reduces the processing time. In a planar scene, all inpainted keyframe pixels are potentially visible from any viewing angle.

In a scene with 3D structure, occluded pixels will occur within the ROI, as the camera moves away from the keyframe; those pixels need to be inpainted (Fig. 2). Our system inserts a new keyframe when the absolute pose difference from the closest keyframe $F(k)$ to the current frame $F(i)$ exceeds a threshold. The difference has been formulated as a Frobenius norm, i.e., $F_{\text{norm}} = ||\mathbf{I} - \mathbf{M}_i^{-1}\mathbf{M}_k||_F$.

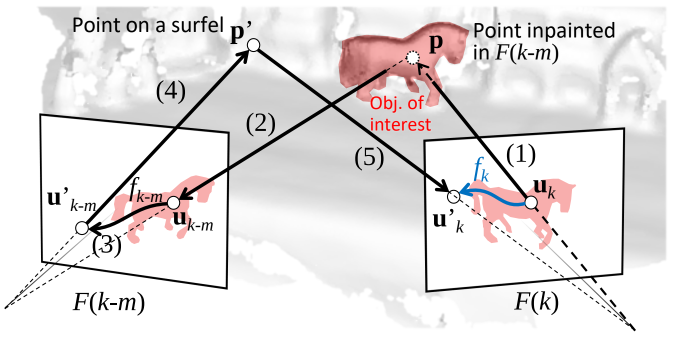


Fig. 5. Keyframe propagation. The mapping f_k (blue arrow) from $\mathbf{u}_k$ to the reference point $\mathbf{u}'_k$ is defined by a series of transformations (black arrows). (1) First, we project $\mathbf{u}_k$ to the world coordinate point $\mathbf{p}$ (2), before we project $\mathbf{p}$ into the other keyframe, which identifies $\mathbf{u}_{k-m}$. (3) At $\mathbf{u}_{k-m}$, we look up f_{k-m} to find $\mathbf{u}'_{k-m}$, and (4) we project $\mathbf{u}'_{k-m}$ back to the world coordinate systems onto $\mathbf{p}'$. (5) Projecting $\mathbf{p}'$ back into the current frame identifies $\mathbf{u}'_k$.

We set the threshold to 0.7 for a Kinect sensor that provides depth maps in meter (Fig. 1).

Additionally, our system checks the number of invalid pixels l', which do not have any labels, i.e., $l' \notin \{L_O, L_{IP}, L_{ROI}\}$. We denote regions of such pixels l' as $T_{l'}$. If $|T_{l'} \bigcap T|/|T| > \epsilon$ for a threshold ϵ (e.g., 0.1), the frame is selected as a new keyframe. In this way, we collect spatially distributed keyframes and safely exclude frames that do not observe target pixels to be inpainted.

3.6 Keyframe propagation

When a keyframe $F(k)$ is inserted, we must ensure that inpainting at $F(k)$ is consistent among previous keyframes $F(k-m)$. We address this requirement by initializing the transformation function f_{k-m} for $F(k-m)$ using the transformation function f_k associated with a keyframe $F(k)$. We start with the closest keyframe, i.e., the keyframe with the minimum absolute pose difference. Fig. 6 shows such an example keyframe propagation.

Since keyframes remain stable over time, reusing the inpainted keyframes in the current frame generates temporally coherent results even for shaky camera motion, as long as the tracking works reliably. To bootstrap the mapping from $F(k-m)$ to $F(k)$, we project $\mathcal{G}$ into $F(k)$ to provide an initial set of inpainted depth values for $\mathbf{M}_k$. Therefore, $F(k)$ contains projections of L_{IP} and L_O in T and L_O in S.

We transform $F(k-m)$ into $F(k)$ via geometric reprojection (i.e., forward warping), with the goal of reusing the pixel mapping stored in f_{k-m} on pixels in T_k. Therefore, we use $\mathcal{G}$ to calculate the transformation of image coordinates $\mathbf{u}_k \in \mathbb{R}^2$ of $F(k)$ into image coordinates $\mathbf{u}_{k-m} \in \mathbb{R}^2$ of $F(k-m)$. These transformations are illustrated in Fig. 5. We start by unprojecting the depth map $\mathcal{D}_k$ to 3D space,

$$\mathbf{p} = \mathbf{K}^{-1}[\mathbf{u}_k^{\mathrm{T}}|1]^{\mathrm{T}}\mathcal{D}_k(\mathbf{u}_k), \tag{2}$$

where $\mathbf{p} = [X, Y, Z]^{\mathrm{T}}$ is a point in the scene, and $\mathbf{K}$ is the 3×3 camera intrinsic matrix. After projecting a pixel into 3D space, the resulting point is further projected into the keyframe $F(k-m)$:

$$\mathbf{u}_{k-m} = \pi([\mathbf{K}|\mathbf{0}]\mathbf{M}_{k-m}\mathbf{M}_k^{-1}[\mathbf{p}^{\mathrm{T}}|1]^{\mathrm{T}}), \tag{3}$$

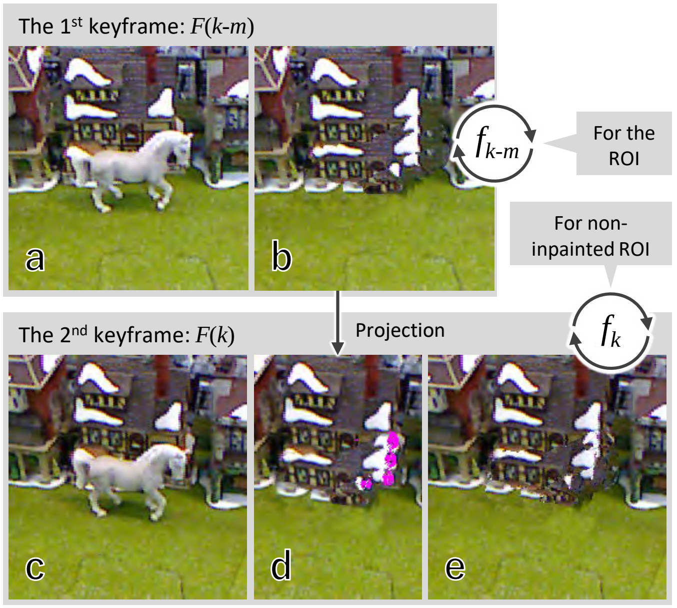


Fig. 6. Spatio-temporally coherent keyframe inpainting. (a) The system selects a keyframe and (b) inpaints the keyframe. (c) Such an inpainted keyframe is projected to the newly inserted keyframe, where (d) non-inpainted regions may be revealed (magenta pixels). (e) Those pixels are newly inpainted to complete the inpainting in the new keyframe.

where $\pi([X, Y, Z]^{\mathrm{T}}) = [X/Z, Y/Z]^{\mathrm{T}}$. Furthermore, we look up the inpainting results within the keyframe $F(k-m)$. Thus, in $F(k-m)$, we use the transformation f_{k-m} at the pixel located at $\mathbf{u}_{k-m}$. The result is the reference position $\mathbf{u}'_{k-m}$ from which the pixel value was taken to inpaint the keyframe.

$$\mathbf{u}'_{k-m} = f_{k-m}(\mathbf{u}_{k-m}) \tag{4}$$

This provides a good guess for color and depth values based on the keyframe data $F(k-m)$. However, to ensure intra-frame consistency, we are interested in selecting pixel values from the new keyframe $F(k)$ rather from the previously preserved keyframe $F(k-m)$. Therefore, we cannot directly take the pixel values at position $\mathbf{u}'_{k-m}$. Instead, we are looking for the corresponding pixel position of $\mathbf{u}'_{k-m}$ in $F(k)$. We compute the projection of $\mathbf{u}'_{k-m}$ into $F(k)$, denoted as $\mathbf{u}'_k$. We derive this transformation by unprojecting the pixel to 3D space, followed by a projection of the corresponding 3D point into $F(k)$.

The complete series of transformations required to map a 2D coordinate $\mathbf{u}_k$ to $\mathbf{u}'_k$ in $F(k)$ is given in Equation 5. Our approach is applied to all pixels within the ROI T_k.

$$f_k : \mathbf{u}_k \overset{Eq.\ 2}{\rightarrow} \mathbf{p} \overset{Eq.\ 3}{\rightarrow} \mathbf{u}_{k-m} \overset{Eq.\ 4}{\rightarrow} \mathbf{u}'_{k-m} \overset{Eq.\ 2}{\rightarrow} \mathbf{p}' \overset{Eq.\ 3}{\rightarrow} \mathbf{u}'_k \tag{5}$$

If a projected point $\mathbf{u}'_k$ is exceeding the bounds of frame $F(k)$, we apply random 2D coordinates to $f_k(\mathbf{u}_k)$. For multiple keyframes, we repeat the above mapping from the closest keyframe to the furthest one, until all the pixels in T_k are processed, or a pre-determined number of keyframes have been processed.

3.7 Keyframe inpainting

Frame pre-processing. There might be missing pixels in a single depth map due to the limitations of the depth sensor, even though the depth map is projected from the global map. Since pixels without depth cannot be used as sources for inpainting, we require sufficiently dense depth to obtain plausible results. Thus, we first fill in missing pixel depth using convolutions [45] for edge-aware inpainting. After this densification, $\mathcal{V}_k$ and $\mathcal{N}_k$ are calculated again from the the depth map.

Finding reference pixels. The initial projection of $\mathcal{G}$ into $F(k)$ may leave some pixels in T_k uninitialized, so we need to fill in these unmapped pixels from scratch. We find the transformation f^* of the remaining pixels by using the PatchMatch algorithm [23] for minimizing the cost function in Equation 6 . Similar to previous examplar-based inpainting, we model the overall cost ρ as a weighted sum of color (texture) similarity, ρ_t, spatial similarity, ρ_s, and a novel geometric similarity term, ρ_g:

$$f^* = \arg\min_f \sum_{\mathbf{u}\in T} w\rho_t(f,\mathbf{u})\rho_g(f,\mathbf{u}) + (1-w)\rho_s(f,\mathbf{u}) \quad (6)$$

The texture cost (Equation 7) minimizes the appearance difference between pixels to be inpainted in T and pixels referenced in $\overline{T}$, while the spatial cost function forces pixels in the area surrounding a target pixel to cluster, so that spatial continuity can be maintained (Equation 8).

$$\rho_t(f,\mathbf{u}) = \sum_{\mathbf{v}\in\{\pm1,\pm2\}^2} ||\mathcal{C}(\mathbf{u}+\mathbf{v}) - \mathcal{C}(f(\mathbf{u})+\mathbf{v})|| \quad (7)$$

$$\rho_s(f,\mathbf{u}) = \sum_{\mathbf{v}\in\{\pm1\}^2} ||f(\mathbf{u})+\mathbf{v} - f(\mathbf{u}+\mathbf{v})|| \quad (8)$$

In addition to texture and spatial similarity, we model a cost function ρ_g for the geometric appearance. In Equation 6, the geometric appearance term modulates the texture similarity ρ_t, acting like a weight that forces both texture and normals to agree. Adding ρ_g as another linear term to sum of ρ_t and ρ_s works as well [9], but it introduces two more additional weighting parameters that must be adjusted. The resulting subtle differences are illustrated in Fig. 12.

Since directly using $\mathcal{D}$ would suffer from the perspective and view-dependent nature of a depth map, we use the normal map for inpainting instead. We derive a normal map from the depth values by using a gradient estimator. However, since the raw depth map suffers from noise and incomplete areas, the derived normal map will be affected as well. Therefore, we use $\mathcal{N}_G$ derived from the projection of the world space depth to the keyframe frame $F(k)$, i.e., $\mathcal{N}_k \rightarrow \mathcal{N}_G$.

$$\rho_g(f,\mathbf{u}) = \sum_{\mathbf{v}\in N_G} 1/\max(\kappa, \mathcal{N}_G(\mathbf{u}+\mathbf{v}) \cdot \mathcal{N}_G(f(\mathbf{u})+\mathbf{v})) \quad (9)$$

Here, κ is a lower bound to avoid division by zero. Pixels having similar normals are clustered naturally. In other words, ρ_g provides a geometrical labeling that limits pixel search to geometrically similar surfaces, overcoming the need for manual labeling used in previous work [9].

The transformation map is randomly initialized when the first keyframe is registered to the system. From the second keyframe on, the closest keyframes' results are propagated as outlined in Section 3.6. We also take a coarse-to-fine approach to find f^* in reasonable time.

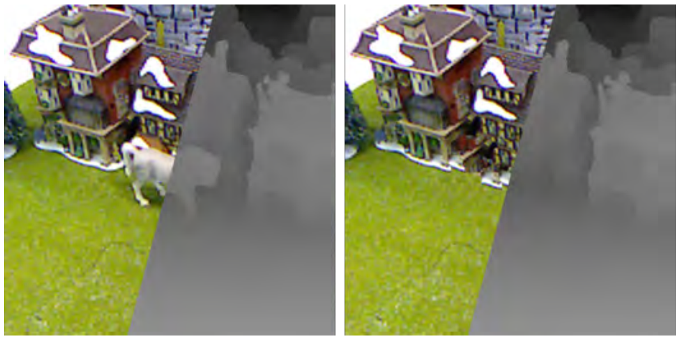


Fig. 7. A complete 3D inpainting for a keyframe. (Left) Given an RGB-D frame, the proposed method finds globally optimized pixels in the color and normal maps calculated from the depth map. (Right) The proposed method also calculates a depth map using sampled depth according to the optimized transformation map.

Generating pixel values. After finding references for all pixels in the ROI, we are able to copy the corresponding pixel values. To inpaint the color $\hat{\mathcal{C}}$, we simply copy the values according to f^*.

$$\hat{\mathcal{C}}(\mathbf{u}) \leftarrow \mathcal{C}(f^*(\mathbf{u})) \quad (10)$$

Since we cannot simply copy view-dependent depth values, we use the normal map $\hat{\mathcal{N}}$ for inpainting 3D structure. The normals at reference pixels can simply be copied like color values.

$$\hat{\mathcal{N}}(\mathbf{u}) \leftarrow \mathcal{N}_G(f^*(\mathbf{u})) \quad (11)$$

For the depth values in the ROI, we compute $\nabla\hat{\mathcal{D}}$, a *gradient field* [46] (depth gradient map) of the sampled depth values. We minimize

$$\min_{\mathbf{u}\in T} \sum (\nabla\hat{\mathcal{D}}^*(\mathbf{u}) - \nabla\hat{\mathcal{D}}(\mathbf{u}))^2 \quad (12)$$

to calculate the inpainted depth map $\hat{\mathcal{D}}^*$. Note that directly sampling pixels from f^* will introduce inconsistencies: Consider a pixel at $\mathbf{u}$ and its right neighbor at $\mathbf{u}+\mathbf{v}$. A naive horizontal gradient

$$\Delta\hat{\mathcal{D}}(\mathbf{u}) = d(f^*(\mathbf{u})) - d(f^*(\mathbf{u})+\mathbf{v}). \quad (13)$$

will usually not match the sampled depth gradient of the adjacent pixel, $d(f^*(\mathbf{u}+\mathbf{v})) - d(f^*(\mathbf{u}+\mathbf{v})-\mathbf{v})$. Therefore, we minimize

$$\min_{\mathbf{u}\in T} \sum (\nabla\hat{\mathcal{D}}^*(\mathbf{u}) - \nabla\hat{\mathcal{E}}(\mathbf{u}))^2, \quad (14)$$

where $\hat{\mathcal{E}}$ is the mean bi-directional depth gradient sample:

$$\begin{aligned}\Delta\hat{\mathcal{E}} =& (d(f^*(\mathbf{u})) - d(f^*(\mathbf{u})+\mathbf{v}) + \\ & d(f^*(\mathbf{u}+\mathbf{v})) - d(f^*(\mathbf{u}+\mathbf{v})-\mathbf{v}))/2\end{aligned} \quad (15)$$

After re-calculating the vertex and normal maps, $\hat{\mathcal{V}}$ and $\hat{\mathcal{N}}$, from the inpainted depth $\hat{\mathcal{D}}^*$, we obtain an inpainted keyframe $\hat{F}(k) = (\hat{\mathcal{C}}_k, \hat{\mathcal{D}}^*_k, \hat{\mathcal{V}}_k, \hat{\mathcal{N}}_k, \mathbf{M}_k)$. Fig. 7 shows an example of an inpainting.

3.8 Keyframe fusion

An inpainted keyframe $\hat{F}(k)$ is fused into the global map to obtain a uniform, consistent representation $\mathcal{G}$. While the user is presented with inpainted frames, the SLAM tracking should not see the inpainting results containing hallucinated data. Therefore, we fuse the inpainted keyframe $\hat{F}(k)$ only with surfels bearing the L_{IP} label, or we insert new surfels labeled L_{IP}, if the unprojected space is vacant. Such newly generated surfels are given a high conficdence value, to ensure that they appear immediately in the next frame. The SLAM tracking only sees surfels with L_O and L_{ROI}.

3.9 View-dependent keyframe blending

Basic blending function. We fill-in the ROI at the current frame $F(i)$ only from completely inpainted keyframes $\hat{F}(k)$. As the surfel resolution in the current view may deviate significantly from the keyframe pixel resolution, calculating a blending weight per surfel can be computationally inefficient, since multiple pixels of $\hat{F}(k)$ can be projected onto a single surfel, or one projected pixel can spread onto multiple surfels.

To avoid the expensive weight calculation at each point of the dense geometry proxy [47], [48], we calculate weights for the M closest keyframes instead and blend the M keyframes with the following weights for projected surfels of label $l(\mathbf{u}) \in L_{IP} \cap T$ and $l(\mathbf{u}) \in L_O \cap T$:

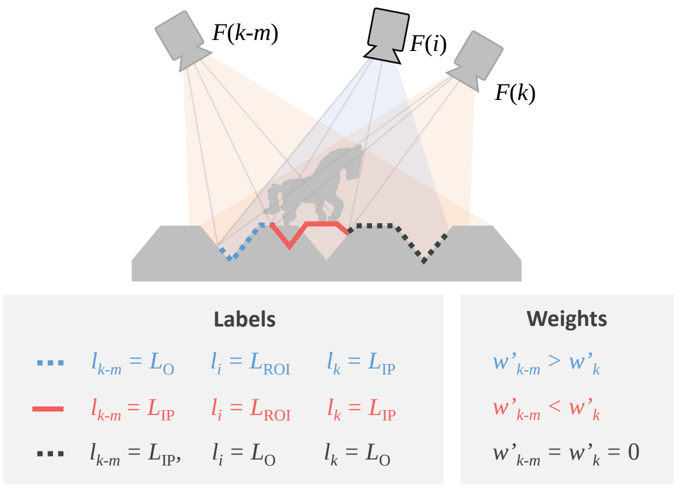
$$w_k = \frac{\exp(-(d_k^{\mathrm{APD}})^2)}{\sum\limits_{m \in M} \exp(-(d_m^{\mathrm{APD}})^2)} \tag{16}$$

Combining observed and inpainted pixels. We always prefer observations over inpainted pixels by giving higher blending weights to such pixels. We distinguish between pixels that have been *observed* before, pixels that have been *inpainted*, and pixels in the *ROI*. As described in Section 3.3, we assign the corresponding labels, L_O, L_{IP}, and L_{ROI}. The label information is already available in each keyframe when it is projected before inpainting. For each pixel $\mathbf{u}$, we check if it is projected to the ROI in a new keyframe $F(k)$:

$$w'_k(\mathbf{u}) = \frac{\exp(-(d_k^{\mathrm{APD}} w_o(\mathbf{u}, 0))^2)}{\sum\limits_{m \in M} \exp(-(d_m^{\mathrm{APD}} w_o(\mathbf{u}, m))^2)} \tag{17}$$
$$w_o(\mathbf{u}, m) = \delta(l_{k-m}(\pi(\mathbf{M}_{k-m}\mathbf{M}_i^{-1}\mathcal{V}_i(\mathbf{u}))), L_{IP}),$$

where δ is the Kronecker delta function. Fig. 8 illustrates our categorization of pixel-wise rendering based on global map labels. First, the red and black regions of $F(i)$ are inpainted in $F(k-m)$, and surfels are labeled as L_{IP}. The blue region remains labeled as observed L_O. Then, the ROI of $F(k)$ is inpainted, i.e., the red and blue regions, using the rendering of $F(k-m)$ in the blue region. However, the label of the blue region is set to L_{IP}, since it is synthetic. Therefore, only the red region is inpainted, receiving information from $F(k-m)$ as the initial guess for the inpainting. When rendering $F(i)$, pixels in the blue and red regions must be inpainted. In our example, we have two keyframes and need to calculate their blending weights.

In the blue region, $w'_{k-m} > w'_k$, even though $F(i)$ is closer to $F(k)$, since $F(k-m)$ is based on a sensor observation. In the red region, we only have surfels with L_{IP} labels. Therefore, the blending weights depend on how close $F(i)$ is to the keyframes, i.e., $w'_{k-m} < w'_k$ in this case. The black region is a direct observation of $F(i)$ and is out of the range of keyframe blending. This procedure combines inpainting-based DR with observation-based DR, while minimizing the inpainting area.



	Labels			Weights
⋯	$l_{k\text{-}m} = L_{\mathrm{O}}$	$l_i = L_{\mathrm{ROI}}$	$l_k = L_{\mathrm{IP}}$	$w'_{k\text{-}m} > w'_k$
━	$l_{k\text{-}m} = L_{\mathrm{IP}}$	$l_i = L_{\mathrm{ROI}}$	$l_k = L_{\mathrm{IP}}$	$w'_{k\text{-}m} < w'_k$
⋯	$l_{k\text{-}m} = L_{\mathrm{IP}},$	$l_i = L_{\mathrm{O}}$	$l_k = L_{\mathrm{O}}$	$w'_{k\text{-}m} = w'_k = 0$

Fig. 8. Distinguishing observed surface and inpainted surface by blending weights in the keyframe blending.

3.10 AR Rendering

Since a full resolution color and geometry map $\hat{F}(i) = (\hat{\mathcal{C}}_i, \hat{\mathcal{D}}_i, \hat{\mathcal{V}}_i, \hat{\mathcal{N}}_i, \mathbf{M}_i)$ from a global map is accessible at each frame, we can use various AR renderings methods for the frame after the inpainting. All the geometry-related maps correspond to G-buffers; therefore, *InpaintFusion* lends itself to any kind of deferred rendering. Fig. 1 shows an example of relighting in the inpainted AR space.

4 Evaluation

We evaluate *InpaintFusion* concerning performance and quality in 3D scenes. Therefore, we implement three different types of geometry proxies on top of *InpaintFusion* to demonstrate how typical geometry proxies used in previous work affect the quality of resulting 3D inpainting, while *InpaintFusion* can maintain its quality in various scenes without explicitly defining geometry proxies.

4.1 Quality assessment

As inpainting has no ground truth, previous work in TABLE 1 does not present quantitative measures and typically relies on the authors' subjective preference. In fact, quantitative assessment in inpainting is an open research problem, as discussed by Isogowa et al. [49]. They propose an automated evaluation for inpainting methods that significantly reduces manual labors in the training step. Such automated quality assessment, however, will not be able to truly reflect human judgement. In addition, spatio-temporal consistency cannot be judged from individual images. For these reasons, we perform a study with human subjects.

The goal of our assessment is to demonstrate that *InpaintFusion* surpasses existing work in terms of subjective quality. To this end, we compared *InpaintFusion* to two other popular approaches, which are based on a geometry proxy. In particular, we compared results of *InpaintFusion* to those generated using a single plane [8], [9], [10], [11], [12] and a multi-plane approach [13], [14]. In the single and multi-plane approach, only one keyframe is inpainted, such that the resulting image looks plausible. Therefore, the quality in subsequent frames only depends on the warping of the inpainted keyframe to the current frame.

To generate the results with the single plane approach, we manually selected three points in the keyframe to define a plane. The geometry for the multi-plane approach is generated using the mean-shift plane estimator proposed by Kawai et al. [13]. For the estimated planes, we generated the corresponding depth map and inpainted the color channels. We estimated the camera pose over time with an RGB-D tracker that minimizes point to plane distances [50] in addition to color [51] residuals in image space, as implemented in OpenCV.

Hereafter, we refer to the contender methods as *Single Plane* and *Multi-Plane*, respectively. Note that these planes inpaint the depth before the color channels are inpainted. Also, such explicit geometry gives enough constraints in Equation 9 to inpaint each plane from geometrically separate pixels without the need for separate inpainting in each plane [13] or manual labeling [9]. For fair comparison in the quality assessment, we used the same tracker and mask images in all three methods including *InpaintFusion*.

4.2 Inpainting quality

Fig. 10 shows inpainting results in selected frames and fusion results of *InpaintFusion* in three scenes that have reasonable 3D structures: *Rock*, *Leaves*, and *Crack*. In *Rock*, we inpainted the handkerchief on a rock. The system generated seamless but fake surface in 3D. Owing to the image-space color and depth inpainting and subsequent fusion of those images, the inpainted structure fits the 3D structure of the real rock even under significant viewport changes. In *Leaves*, we inpainted the dead leaf among the green leaves to replace the dead leaf with green leaves hallucinated by the proposed system. Note that real leaves are occluded by the generated leaves, and partially observable real leaves from different viewpoints retain their shape and color in each view. In *Crack*, we inpainted a crack on a rock to virtually fix the crack. Note how the inpainting kept the geometric edge of two surfaces.

We also compare inpainted color and depth maps of all types of geometry proxies. Here, for lack of space, we show results of only the *Crack* dataset as a typical case (Fig. 11). More results are provided in the accompanying video. Note that all inpainting results have a different appearance in each trial due to the randomized initial transfer map and the different geometry proxy. Besides, none of the methods in TABLE 1 explicitly provide depth, although approaches that use AR markers or SLAM as a tracker coud provide depth from the marker position or SLAM points. Even though single frame appearance may be plausible, in motion, wrongly fitted planes reveal the ROI due to the inconsistent disparities. This effect is best observed in the accompanying video.

We specified three points on the top surface for *Single Plane*, defining an infinite plane. Consequently, the inpainted region on the side surface floats when the camera moves. *Multi-Plane* estimated two dominant planes for the top and the side surfaces. Nevertheless, this method always selects the closest plane distance from the camera; the shape of the scene resembles a concave wall-and-floor geometry. This geometry is not correctly representing the scene, leading to inconsistent inpainting results in all views except at the keyframe. In contrast, *InpaintFusion* plausibly estimates two surfaces that fit the real geometry well, resulting in seamless and consistent inpainting from various viewpoints (Fig. 10).

4.3 User Study

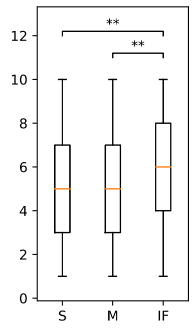
Design We designed a repeated measures within-subjects study to compare the inpainting quality of different inpainting methods. Therefore, we introduced the indepenedent variable ”inpainting method” with three conditions: *Single Plane* (S), *Multi-Plane* (M), and *InpaintFusion* (IF). As dependent variables, we collected ratings for image and video results, s_{I} and s_{V}, respectively. To analyze how different geometry proxies impact inpainting results under camera motions, we also calculate the differences, $s_{\mathrm{V}} - s_{\mathrm{I}}$ scores.

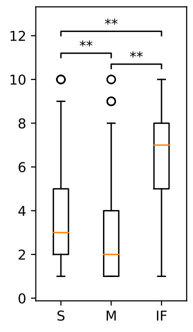
Task We designed a task for rating image and video inpainting results on a 10-point Likert scale, using nine scenes including the scenes in Fig. 10 and Fig. 14. Provided textual information, the participants were instructed to understand the purpose of the inpainting process in each scene, e.g., the inpainting is used to hide logos in the scene. Later we asked the participants to evaluate how well each inpainting method achieved the purpose[1]. For image results, we chose a corresponding frames from each of the videos showing the different inpainting methods.

Apparatus We used a web-based survey system, SurveyMonkey, to collect responses from people of different expertise and nationalities, after email invitation. The participants were instructed to use at least a 13” screen to ensure reasonable viewing conditions.

Procedure After receiving textual instructions and signing an informed consent form, participants evaluated a series of inpainted images, followed by evaluation of inpainting videos. In each rating, three still images or videos were presented side-by-side, showing the original with and without the highlighted interest region and the inpainted image. 55 participants (six female, age $\bar{X}$ = 31.7, SD = 7.8 years old) volunteered for the study. On a scale from one to five, where five means the best, the mean self-rated experience concerning inpainting was 2.4 (SD= 1.1). The participants scored the nine inpainting test cases in random order. A session took approximately 17 minutes. With 55 participants, nine repetitions and three inpainting methods, we collected a total of $55 \times 9 \times 3 = 1,485$ ratings for image and the same number of ratings for video results.

1. For further details, we provide a supplemental videos.


(a) s_{I}


(b) s_{V}

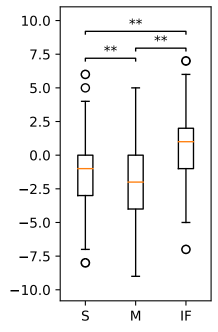

(c) $s_{\mathrm{V}} - s_{\mathrm{I}}$

Fig. 9. Study results in image evaluation (a), video evaluation (b), and deteriorations from image to video (c).

Hypotheses We did not expect significant differences in still image results where no motion disparities appear (**H1**). However, due to the fused 3D geometry proxy of the proposed method, we expected IF to have significantly higher scores in video results than S and M (**H2**). Moreover, we expected IF to have significantly fewer deteriorations of the $s_{\mathrm{V}} - s_{\mathrm{I}}$ score than S and M (**H3**).

Results The score data was analyzed using a non-parametric Friedman test followed by pairwise Wilcoxon signed rank tests with Bonferroni correction. The reported p-values have been Bonferroni corrected to reflect a significance level of 0.05. The statistical analysis was performed using R software.

Friedman tests revealed significant differences in image results (χ^2(2)=852.02, p<0.001), in video results (χ^2(2)=1018.00, p<0.001), and in $s_{\mathrm{V}} - s_{\mathrm{I}}$ scores (χ^2(2)=1018.00, p<0.001). Post-hoc tests indicated that IF scores (Mdn=6) were statistically higher than S scores (Mdn=5, Z=-5.540, p<0.001, r=0.249) and M scores (Mdn=5, Z=-4.66, p<0.001, r=0.209) in image results. Post-hoc tests indicated that all combinations in video results have significant differences; IF scores (Med=7) were statistically higher than S scores (Mdn=3, Z=-16.589, p<0.001, r=0.746) and M scores (Mdn=2, Z=-17.775, p<0.001, r=0.799), and S scores were statistically higher than M scores (Z=-7.342, p<0.001, r=0.330). Also, in $s_{\mathrm{V}} - s_{\mathrm{I}}$ scores, post-hoc tests indicated that all combinations have significant differences; IF scores (Med=1) were statistically higher than S scores (Mdn=−1, Z=-15.071, p<0.001, r=0.677) and M scores (Mdn=−2, Z=-16.251, p<0.001, r=0.730), and S scores were statistically higher than M scores (Z=-6.366, p<0.001, r=0.286). Fig. 9 summarizes the study results.

Discussion IF is scored one unit higher in the median than the others in image results. Therefore, we reject our pessimistic hypothesis H1, as IF performed better than expected. One possible explanation could be that the geometry term of IF constrains the search range to a proper region, while S must search pixels only using colors. We observed that M tends to leak colors in wrong plane regions, when it fails to estimate planes correctly.

For video, IF scored even one unit higher than for still images, clearly outperforming both S and M. We explain this by the well-maintained temporal and spatial coherence of IF under 6DOF motions. IF gave the participants better impressions than in image results, while scores for S and M were lowered due to geometrical misalignments. S repeatedly failed when there were multiple objects of interest at different depths or when the region had varying depth. Also, we observed that, when M fails to estimate the right planes, mismatched disparities in the video are created. Fig. 11 shows such typical cases. Overall, we accept our optimistic hypotheses H2 and H3. We conclude that IF can maintain the quality even in scenes where planar inpainting approaches fail.

TABLE 2
Average time (ms) spent in each stage of the main thread.

Component	Runtime [ms]
Tracking (OpenCV / GPU ICP)	128.16 / 21.44
SLAM fusion	4.60
Inpainted KF fusion	8.00 / KF
View-dependent KF blending	1.93
Misc (Frame pre-processing & data handling)	3.20
Total (OpenCV / GPU ICP)	137.88 / 31.16

TABLE 3
Average time spent in each stage of the inpainting thread, which runs in parallel to the main thread and therefore does not stall the application.

Component	Runtime [ms]
KF propagation	19.75
Transformation map optimization (50 raster-scans at each of a six level pyramid)	4288.33
Depth estimation from depth samples	96.67
Mask ratio	17.02 %
Total	4385.00

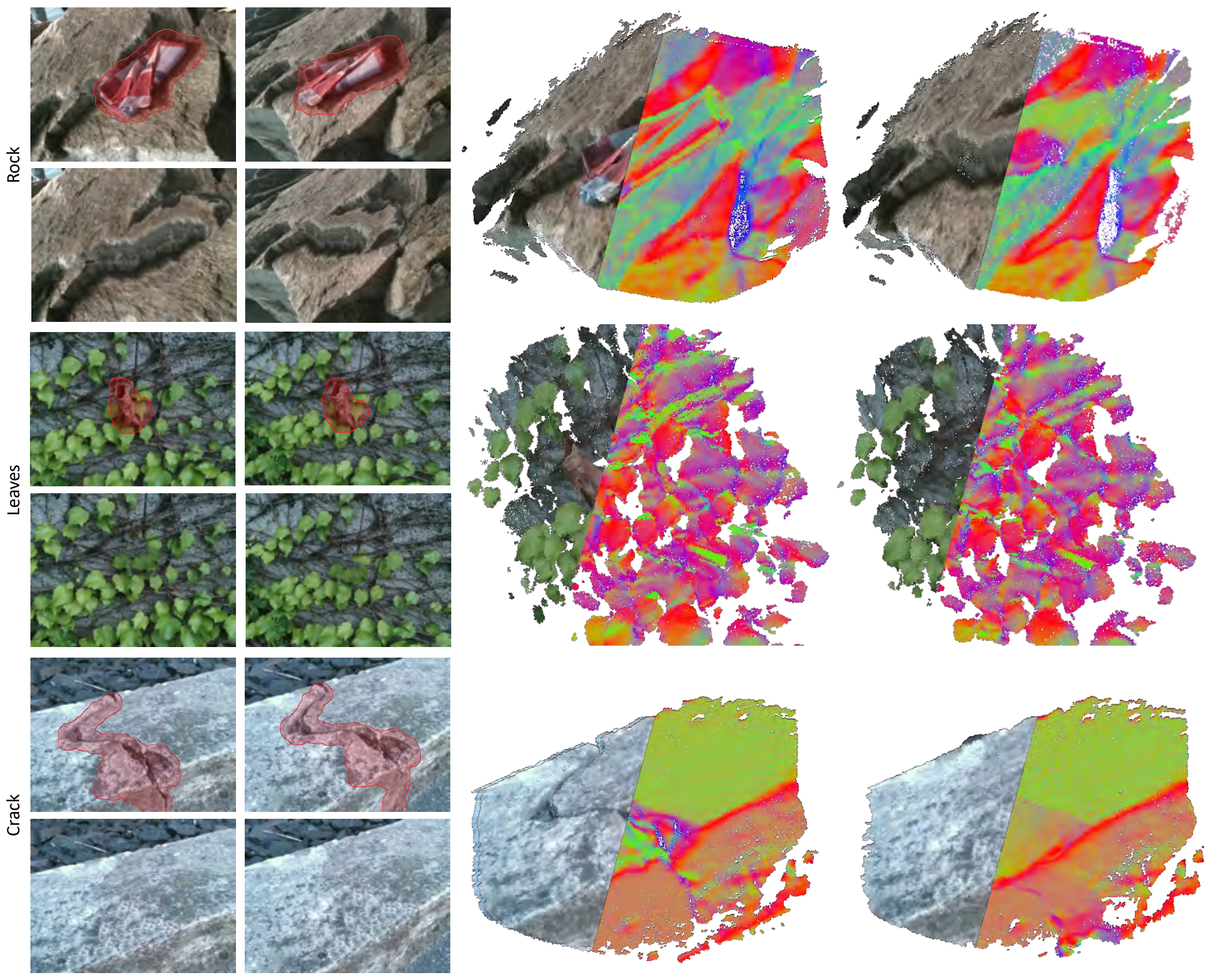
4.4 Runtime performance

We implemented *InpaintFusion* on a notebook computer (Intel Core i7-6567U with 3.3 GHz, 16 GB RAM, external NVIDIA GeForce GTX1080Ti connected via Thunderbolt 3) running Windows 10. As RGB-D sensor, we either used a Microsoft Kinect v1 or an Intel Realsense SR300, running at 30Hz in 640×480 resolution. We implemented our system in two threads, one performing keyframe inpainting and another one for the rest of the processing, including GPU tasks (using OpenGL and GLSL), SLAM, rendering, and passing selected keyframes to the inpainting thread.

TABLE 2 and 3 summarize the performance in milliseconds in the main and inpainting threads, respectively. Overall, *InpaintFusion* operates approximately at 31.16Hz. Although the first keyframe inpainting is most expensive and finishes in 4.4 sec., it runs in the background without interfering with tracking, and it takes less in the following keyframes due to keyframe propagation. In comparison with existing methods using a similar hardware setup, *InpaintFusion* performs equal or even faster, even though it is capable of full 3D inpainting that has never been achieved before (please see the accompanying video).

5 Limitations and Future Work

While *InpaintFusion* generalizes the inpainting-based DR methods, some points need to be addressed to further improve the quality.



(a) Raw images w/ ROI (top) our results (bottom) (b) Color and normal maps of raw image data (left) and results after inpainting (right)

Fig. 10. *InpaintFusion* in three different scenes.

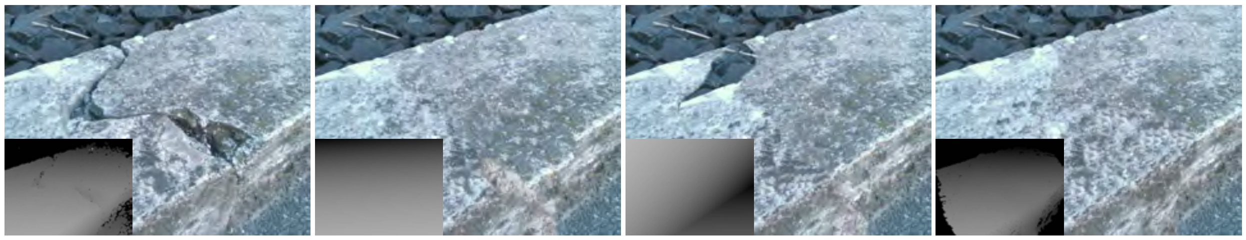


Fig. 11. Comparison of different types of geometry proxies on the *Crack* dataset. From left to right: original, single plane, multi-plane, and *InpaintFusion*. The insets in the lower left corner show depth encoded as greyscale values.

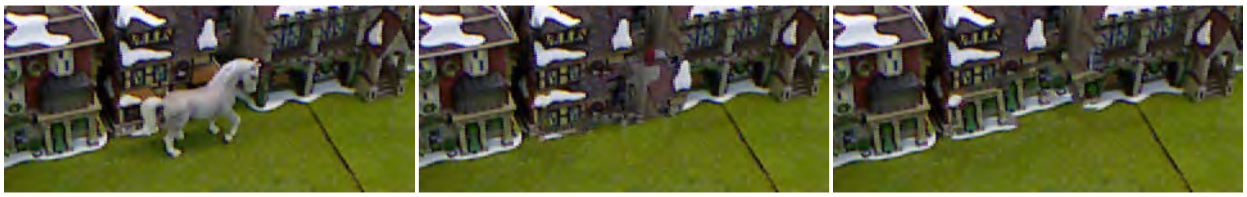


Fig. 12. Comparison of similarity metrics. From left to right: original, linear combination of texture and geometry similarity, and multiplicative combination of texture and geometry similarity (our approach). The linear combination requires three parameters to be adjusted for inpainting, while the multiplicative approach requires one parameter.

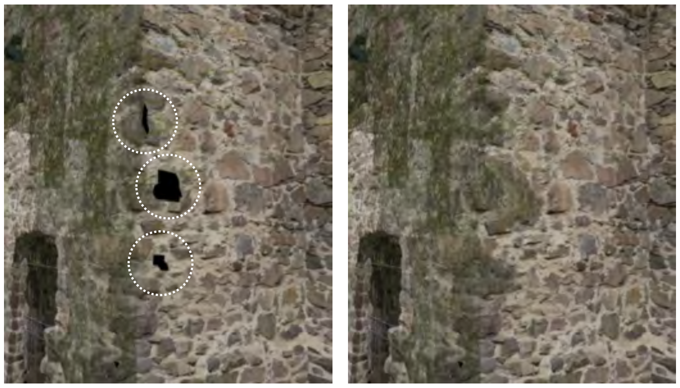


Fig. 13. Scene completion using *InpaintFusion*. Our approach is able to complete 3D reconstructions. In this example, it completes a reconstruction of the CityWall dataset provided by TU Darmstadt[2]. *InpaintFusion* generates the missing geometry and color information (right), which is highlighted by white circles in the image on the (left).

Integer space transformation map propagation Transformation map f represents pixel-to-pixel offsets, i.e., $f \in \mathbb{Z}$. Therefore, propagating such a map to the next keyframe leads to nearest-neighbor interpolation in image space. Such an approach is prone to aliasing, as seen in $\mathcal{C}_k(f_k^*(\mathbf{u}))$ of Fig. 6 (e). A potential solution to this problem in our system is to set fairly large thresholds discussed in Section 3.5. We could re-optimize the propagated transformation map using the warped color map from the closest keyframe as a strong constraints in the appearance costs minimization [9]. However, once the optimization finds better pixel-to-pixel relationships than the current ones, the resulting appearance of the inpainting will differ substantially from the other keyframes. Planar inpainting avoids this problem, since it does not have to handle occlusions. To mitigate the aliasing problems, we can use view-dependent keyframe blending, as described in Section 3.9, to compose multiple keyframes to render the current inpainted region.

Occlusion handling with real and inpainted depths In case our system reconstructs a potentially observable background, and the user perfectly labels an object of interest before inpainting starts, our system can safely project the reconstructed surfels belonging to L_O to exclude ones with L_{ROI} from inpainting. However, in practice, the surfel-wise colors are not precise enough to fill in L_{ROI} as they are. This issue is demonstrated in an existing attempt [43], and surfels with view-dependent properties could mitigate this problem [52]. Therefore, we chose to inpaint the entire ROI in the first keyframe. This means that, in case a real background is observed after inpainting, that inpainted depth and real depth may disagree, leading to discontinuities at the ROI border. To minimize such discontinuities, we can use automatic segmentation, using the manually specified ROI as seed, to obtain a better labeling that supresses problems at borders.

Bundle inpainting using all keyframes Adding more constraints in the transformation map optimization will lead to more robust inpainting, but it will also further restrict the pixel search range. This can lead to a lack of pixel sources. One could search pixels in all preserved keyframes to optimize a single transformation map, but this would require another term in the optimization to represent pixel continuities across keyframes. One good example we could find projects multi-view images to common planes to use available pixel sources for the inpainting [53], although the authors stress the difficulties to apply the strategy for non-planar regions. We find such an extension an interesting avenue of future research.

2. https://www.gcc.tu-darmstadt.de/home/proj/mve/

6 Conclusion

This paper presents a novel approach for interactive image inpainting in 3D. We have shown how the integration of fusion and multi-keyframe inpainting delivers globally consistent and appealing results. Our system ensures frame-to-frame coherence of the inpainted results by considering a 3D geometric term in addition to texture in image space. This ability improves the range of possible use cases for interactive DR applications, for instance, we can target multi-view rendering for stereoscopic display devices. Furthermore, our system supports image editing by its ability to add 3D visual effects to inpainted images. This enables quickly adding 3D visual effects to images and videos, providing a tool for previewing image and video editing operations.

InpaintFusion also opens up possibilities for AR effects after the inpainting. For example, we demonstrated relighting from virtual car headlights and physical animation of snowballs in the inpainted region (Fig. 1). We also made real objects virtually interactive by replacing the real object with a scanned model after removing the real object. In case the user could scan the backgrounds in advance, our system can erase frontally occluding objects for X-ray vision. In this case, surfels belong to the ROI are replaced with observed ones. *InpaintFusion* can also supply RGB-D inpainting for scene completion, filling holes in a point cloud corresponding to unobserved areas in the scene or reconstruction failures (Fig. 13). Prototypes of these applications are presented in the supplementary materials.

Acknowledgments

This work was enabled by the FFG (grant 859208) and the Austrian Science Fund (grant P30694).

References

[1] S. Mori, S. Ikeda, and H. Saito, "A survey of diminished reality: Techniques for visually concealing, eliminating, and seeing through real objects," *IPSJ Trans. on Computer Vision and Applications*, vol. 9, no. 17, 2017.
[2] D. Schmalstieg and T. Höllerer, *Augmented Reality: Principles and Practice*. Addison-Wesley Professional, 2016.
[3] F. Cosco, C. Garre, F. Bruno, M. Muzzupappa, and M. A. Otaduy, "Visuo-haptic mixed reality with unobstructed tool-hand integration," *IEEE Trans. on Visualization and Computer Graphics*, vol. 19, no. 1, pp. 159–172, 2013.
[4] Z. Li, Y. Wang, J. Guo, L.-F. Cheong, and S. Z. Zhou, "Diminished reality using appearance and 3d geometry of internet photo collections," in *Proc. IEEE Int. Symp. on Mixed and Augmented Reality*, 2013, pp. 11—19.
[5] P. Barnum, Y. Sheikh, A. Datta, and T. Kanade, "Dynamic seethroughs: Synthesizing hidden views of moving objects," in *Proc. IEEE Int. Symp. on Mixed and Augmented Reality*, 2009, pp. 111—114.

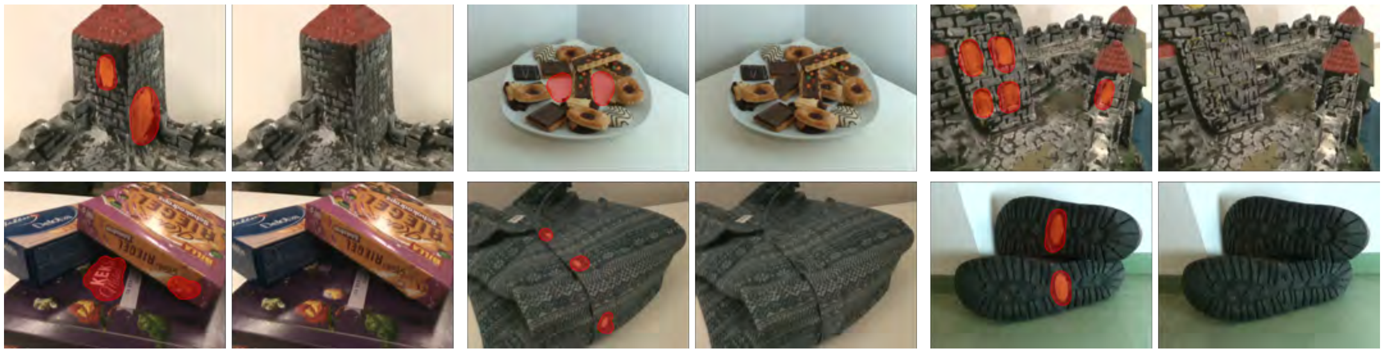


Fig. 14. Additional results. Raw images w/ ROI (left) our results (right). As we discuss in Section 4.3, static images do not clearly show the advantages of our method. Therefore, we strongly recommend readers to watch the results in motion in the provided supplemental videos.

[6] S. Zokai, J. Esteve, Y. Genc, and N. Navab, "Multiview paraperspective projection model for diminished reality," in *Proc. IEEE Int. Symp. on Mixed and Augmented Reality*, 2003, pp. 217—-226.

[7] F. Rameau, H. Ha, K. Joo, J. Choi, K. Park, and I. Kweon, "A real-time augmented reality system to see-through car," *IEEE Trans. on Visualization and Computer Graphics*, vol. 22, no. 11, 2016.

[8] N. Kawai, T. Sato, Y. Nakashima, and N. Yokoya, "Augmented reality marker hiding with texture deformation," *IEEE Trans. on Visualization and Computer Graphics*, vol. 23, no. 10, pp. 2288–2300, 2017.

[9] J. Herling and W. Broll, "High-quality real-time video inpainting with pixmix," *IEEE Trans. on Visualization and Computer Graphics*, vol. 20, no. 6, pp. 866–879, 2014.

[10] S. Siltanen, "Texture generation over the marker area," in *Proc. IEEE Int. Symp. on Mixed and Augmented Reality*, 2006, pp. 253–254.

[11] O. Korkalo, M. Aittala, and S. Siltanen, "Light-weight marker hiding for augmented reality," in *Proc. IEEE Int. Symp. on Mixed and Augmented Reality*, 2010, pp. 247–248.

[12] J. Herling and W. Broll, "Advanced self-contained object removal for realizing real-time diminished reality in unconstrained environments," in *Proc. IEEE Int. Symp. on Mixed and Augmented Reality*, 2010, pp. 207–212.

[13] N. Kawai, T. Sato, and N. Yokoya, "Diminished reality based on image inpainting considering background geometry," *IEEE Trans. on Visualization and Computer Graphics*, vol. 22, no. 3, pp. 1236–1247, 2016.

[14] S. Siltanen, "Diminished reality for augmented reality interior design," *The Visual Computer*, vol. 33, pp. 193–208, 2015.

[15] S. Meerits and H. Saito, "Real-time diminished reality for dynamic scenes," in *ISMAR Workshop on Diminished Reality*, 2015, pp. 53––59.

[16] R. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2003.

[17] S. Ilan and A. Shamir, "A survey on data-driven video completion," *Computer Graphics Forum*, vol. 34, no. 6, 2015.

[18] J. Y. A. Wang and E. H. Adelson, "Representing moving images with layers," *IEEE Trans. on Image Processing*, vol. 3, no. 5, pp. 625–638, 1994.

[19] V. Lepetit, M.-o. Berger, and L.-i. Lorraine, "An intuitive tool for outlining objects in video sequences: Applications to augmented and diminished reality," in *Proc. IEEE Int. Symp. on Mixed and Augmented Reality*, 2001, pp. 159–160.

[20] Y. Shen, F. Lu, X. Cao, and H. Foroosh, "Video completion for perspective camera under constrained motion," in *Proc. Int. Conf. on Pattern Recognition*, 2006, pp. 63–66.

[21] F. Klose, O. Wang, J.-C. Bazin, M. Magnor, and A. Sorkine-Hornung, "Sampling based scene-space video processing," *ACM Trans. on Graphics (Proc. SIGGRAPH)*, vol. 34, no. 4, 2015.

[22] A. Criminisi, P. Perez, and K. Toyama, "Region filling and object removal by exemplar-based image inpainting," *IEEE Trans. on Image Processing*, vol. 13, no. 9, pp. 1200–1212, 2004.

[23] C. Barnes, E. Shechtman, A. Finkelstein, and D. B. Goldman, "Patchmatch: A randomized correspondence algorithm for structural image editing," *ACM Trans. on Graphics (Proc. SIGGRAPH)*, vol. 28, no. 3, 2009.

[24] C. Barnes, E. Shechtman, D. B. Goldman, and A. Finkelstein, "The generalized patchmatch correspondence algorithm," in *European Conf. on Computer Vision*, 2010, pp. 29–43.

[25] C. Kunert, T. Schwandt, and W. Broll, "An efficient diminished reality approach using real-time surface reconstruction," in *Cyberworlds*, 2019.

[26] P. Debevec, "Rendering synthetic objects into real scenes: Bridging traditional and image-based graphics with global illumination and high dynamic range photography," in *Proc. Annual Conf. on Computer Graphics and Interactive Techniques (SIGGRAPH)*, 1998, pp. 189–198.

[27] K. Rohmer, W. Büschel, R. Dachselt, and T. Grosch, "Interactive near-field illumination for photorealistic augmented reality with varying materials on mobile devices," in *Proc. IEEE Int. Symp. on Mixed and Augmented Reality*, 2015, pp. 1349–1362.

[28] C. Fehn, "Depth-image-based rendering (dibr), compression, and transmission for a new approach on 3d-tv," in *Stereoscopic Displays and Virtual Reality Systems XI.*, vol. 5291, no. 93. Proc. Int. Society for Optics and Photonics, 2004, p. 1.

[29] A. Holynski and J. Kopf, "Fast depth densification for occlusion-aware augmented reality," *ACM Trans. on Graphics*, vol. 37, no. 6, pp. 194:1–194:11, Dec. 2018.

[30] J. Valentin, A. Kowdle, J. T. Barron, N. Wadhwa, M. Dzitsiuk, M. Schoenberg, V. Verma, A. Csaszar, E. Turner, I. Dryanovski, J. Afonso, J. Pascoal, K. Tsotsos, M. Leung, M. Schmidt, O. Guleryuz, S. Khamis, V. Tankovitch, S. Fanello, S. Izadi, and C. Rhemann, "Depth from motion for smartphone ar," *ACM Trans. on Graphics*, vol. 37, no. 6, pp. 193:1–193:19, Dec. 2018.

[31] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial networks," in *Proc. Advances in Neural Information Processing Systems*, 2014, pp. 2672–2680.

[32] S. Iizuka, E. Simo-Serra, and H. Ishikawa, "Globally and locally consistent image completion," *ACM Trans. on Graphics (Proc. SIGGRAPH)*, vol. 36, no. 4, pp. 107:1–107:14, 2017.

[33] D. Pathak, P. Krähenbühl, J. Donahue, and T. Darrell, "Context encoders: Feature learning by inpainting," in *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, 2016, pp. 2536–2544.

[34] R. A. Yeh, C. Chen, T. Y. Lim, A. G. Schwing, M. Hasegawa-Johnson, and M. N. Do, "Semantic image inpainting with deep generative models," in *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, 2017, pp. 6882–6890.

[35] J. Yu, Z. Lin, J. Yang, X. Shen, X. Lu, and T. S. Huang, "Generative image inpainting with contextual attention," in *Proc. IEEE/CVF Conf. on Computer Vision and Pattern Recognition*, 2018.

[36] I. Laina, C. Rupprecht, V. Belagiannis, F. Tombari, and N. Navab, "Deeper depth prediction with fully convolutional residual networks," in *Proc. IEEE Int. Conf. on 3D Vision*, 2016, pp. 239–248.

[37] C. Godard, O. Mac Aodha, and G. J. Brostow, "Unsupervised monocular depth estimation with left-right consistency," in *Proc. IEEE/CVF Conf. on Computer Vision and Pattern Recognition*, 2017.

[38] H. Dhamo, K. Tateno, I. Laina, N. Navab, and F. Tombari, "Peeking behind objects: Layered depth prediction from a single image," *Pattern Recognition Letters*, vol. 125, pp. 333 – 340, 2019.

[39] M. Keller, D. Lefloch, M. Lambers, S. Izadi, T. Weyrich, and A. Kolb, "Real-time 3d reconstruction in dynamic scenes using point-based fusion," in *Proc. Int. Conf. on 3D Vision*, 2013, pp. 1–8.

[40] H. Pfister, M. Zwicker, J. van Baar, and M. Gross, "Surfels: Surface elements as rendering primitives," in *Proc. SIGGRAPH*, 2000, pp. 335–342.

[41] C. Tomasi and R. Manduchi, "Bilateral filtering for gray and color images," in *Proc. Int. Conf. on Computer Vision*, 1998, pp. 839–846.

[42] K.-L. Low, "Linear least-squares optimization for point-to-plane icp surface registration," Dept. of Computer Science, University of North Carolina, Chapel Hill, Tech. Rep. TR04-004, 2004.

[43] Y. Nakajima, S. Mori, and H. Saito, "Semantic object selection and detection for diminished reality based on slam with viewpoint class," in *2017 IEEE International Symposium on Mixed and Augmented Reality (ISMAR-Adjunct)*, 2017, pp. 338–343.

[44] C. Li, H. Xiao, K. Tateno, F. Tombari, N. Navab, and G. D. Hager, "Incremental scene understanding on dense slam," in *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, October 2016.

[45] T. Schöps, M. R. Oswald, P. Speciale, S. Yang, and M. Pollefeys, "Real-time view correction for mobile devices," *IEEE Trans. on Visualization and Computer Graphics*, vol. 23, no. 11, pp. 2455–2462, 2017.

[46] P. Pérez, M. Gangnet, and A. Blake, "Poisson image editing," *ACM Trans. on Graphics (Proc. SIGGRAPH)*, vol. 22, no. 3, pp. 313–318, 2003.

[47] C. Buehler, M. Bosse, L. McMillan, S. Gortler, and M. Cohen, "Unstructured lumigraph rendering," in *Proc. Annual Conf. on Computer Graphics and Interactive Techniques (SIGGRAPH)*, 2001, pp. 425–432.

[48] A. Davis, M. Levoy, and F. Durand, "Unstructured light fields," *Computer Graphics Forum*, vol. 31, no. 2, pp. 305–314, 2012.

[49] M. Isogawa, D. Mikami, K. Takahashi, D. Iwai, K. Sato, and H. Kimata, "Which is the Better Inpainted Image? Training Data Generation Without Any Manual Operations," *Int. Journal of Computer Vision*, 2018.

[50] R. A. Newcombe, S. Izadi, O. Hilliges, D. Molyneaux, D. Kim, A. J. Davison, P. Kohli, J. Shotton, S. Hodges, and A. Fitzgibbon, "Kinectfusion: Real-time dense surface mapping and tracking," in *Proc. IEEE Int. Symp. on Mixed and Augmented Reality*, 2011, pp. 127–136.

[51] F. Steinbrücker, J. Sturm, and D. Cremers, "Real-time visual odometry from dense rgb-d images," in *ICCV Workshops*. IEEE Computer Society, 2011, pp. 719–722.

[52] J. J. Park, R. Newcombe, and S. Seitz, "Surface light field fusion," in *2018 International Conference on 3D Vision (3DV)*. IEEE, 2018, pp. 12–21.

[53] J. Philip and G. Drettakis, "Plane-based multi-view inpainting for image-based rendering in large scenes," in *Proc. of the ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games*, may 2018.

Shohei Mori is a postdoctoral researcher at the Institute of Computer Graphics and Vision (ICG) at Graz University of Technology. His focus lies in augmented and diminished reality and the related computer vision and display technologies. Shohei received his B.S. (2011), M.S. (2013), and PhD (2016) in engineering at Ritsumeikan University, Japan. Before joining the ICG, he worked as a Research Fellowship for Young Scientists (DC-1) from the Japan Society for the Promotion of Science (JSPS) during his Ph.D degree, and he started his career as a JSPS Research Fellowship for Young Scientists (PD) at Keio University, Japan.

Okan Erat is a research assistant at Institute of Computer Graphics and Vision at the Technical University of Graz (TU-Graz). Before joining TU-Graz He studied Biomedical Computing at Technical University of Munich (TUM). His focus lies on image based rendering for telepresence systems.

Wolfgang Broll is a full professor at Ilmenau University of Technology, heading the Virtual Worlds and Digital Games group. He received his Master's (Dipl.-Inf.) in Computer Science at Darmstadt University of Technology (1993) and PhD in Computer Science at Tübingen University (1998). He was a lecturer at RWTH Aachen from 2000 to 2009. From 1994 to 2012 he was heading the VR/AR activities at Fraunhofer FIT in Sankt Augustin. He was also a co-founders and manager of fayteq. He is an ACM SIGGRAPH pioneer member, member of IEEE Computer Society and Germany's computer society (GI). He is currently concerned with augmented reality related technologies, including diminished and mediated reality.

Hideo Saito received the Ph.D. degree in electrical engineering at Keio University, Japan, in 1992. Since 1992, he has been on the Faculty of Science and Technology, Keio University. From 1997 to 1999, he joined the Virtualized Reality Project at the Robotics Institute, Carnegie Mellon University, as a Visiting Researcher. His recent activities in academic conferences include being a Program Chair of ACCV 2014, a General Chair of ISMAR 2015, and a Program Chair of ISMAR 2016 and EuroVR 2020.

Dieter Schmalstieg is a Professor at Graz University of Technology, Austria. His research interests are augmented reality, virtual reality, computer graphics, visualization and human-computer interaction. He received a PhD from Vienna University of Technology.

Denis Kalkofen is an Assistant Professor at the Institute of Computer Graphics and Vision (ICG) at Graz University of Technology, Austria. Before joining ICG, he was a member of the Virtual Reality Laboratory at University of Michigan. In 2019, he joined the Wearable Computer Laboratory at the University of South Australia as Visiting Researcher and the Computational Imaging Laboratory at Stanford University as Visiting Assistant Professor. His research is focused on visual computing for developing visualization, interaction, display and authoring techniques for Mixed Reality environments.