Fast Intra-Frame Video Splicing for Occlusion Removal in Diminished Reality

Chengyuan Lin$^{[0000-0002-5768-9000]}$ and Voicu Popescu$^{[0000-0002-8767-8724]}$

Purdue University, West Lafayette IN 47906, USA
popescu@purdue.edu

Abstract. In a real world scene objects of interest might be occluded by other objects. Diminished reality (DR) aims to remove such occluders. A popular approach is to acquire the geometry of the occluded scene, and then to render it from the user's viewpoint, effectively erasing the occluder. However, the approach is ill-suited for scenes with intricate and dynamic geometry, which cannot be acquired quickly, completely, and with only minimal equipment. This paper proposes a method to erase an occluder in a primary video by splicing in pixels from a secondary video. For each frame, the method finds the region in the secondary frame that corresponds to the occluder shadow, and integrates it seamlessly into the primary frame. Precise matching of the occluder contour is achieved by a novel pipeline with a tracking, global alignment, and local alignment stages. The result is a continuous multiperspective frame, which shows most of the scene from the primary viewpoint, except for the part hidden by the occluder, which is shown from the secondary viewpoint. A high quality multiperspective transparency effect is achieved for complex scenes, without the high cost of 3D acquisition. When compared with other DR methods, the proposed method shows fewer artifacts and better continuity.

Keywords: Occluder removal · Video splicing · Multiperspective visualization · Diminished reality · Augmented reality

1 Introduction

Augmented reality improves a user's view of the real world by *adding* graphical annotations. However, improving the user's view sometimes calls for *removing* objects from the user's view. Such diminished reality visualizations are used to eliminate distracting clutter, to preview possible changes to a real world scene without actually modifying it, and to let the user see hidden parts of the scene quickly, from the current location. Removing an occluder from the user's view of the real world requires finding the footprint of the occluder, finding a visual description of what the user should see in the absence of the occluder, and transferring it to the occluder footprint. One approach is 3D scene acquisition. Once the geometry of the scene is known, the scene can be rendered from the

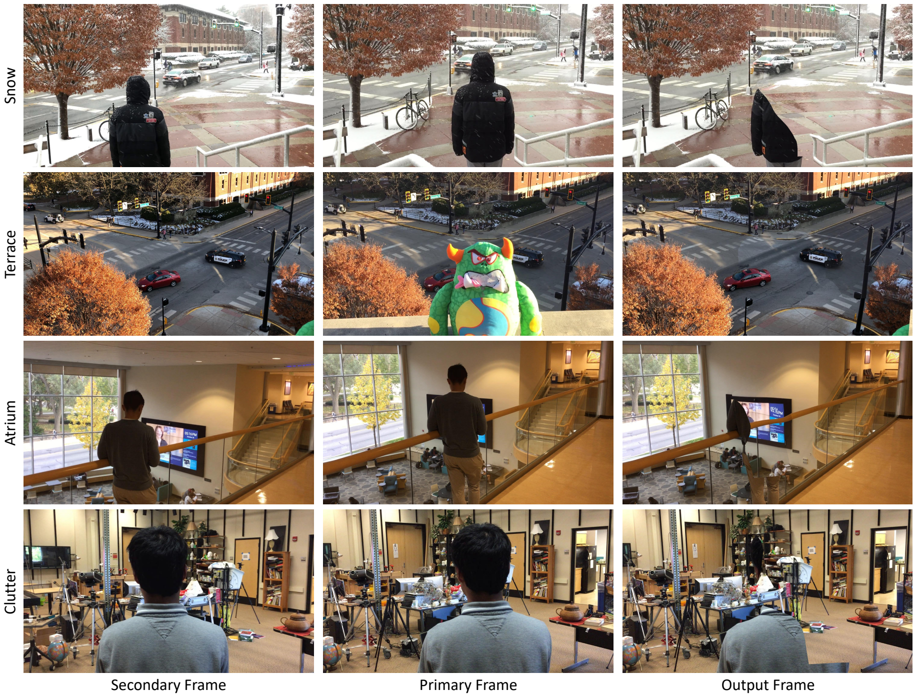

Fig. 1. Results preview. The output frame is obtained by erasing the occluder from the primary frame using pixels from the secondary frame. The result is a quality transparency effect, with good continuity at the occluder contour. Our method supports intricate dynamic scenes, and the frame rate is 75fps or better for an output resolution of 1,920 × 1,080.

user's viewpoint, without the occluder. This works well, for example, when one wants to remove an object from a corner of a room, whose color and geometry are easy to acquire or synthesize. A challenging case for this approach is when the parts of the scene hidden by the occluder have intricate and dynamic appearance and geometry, which cannot be acquired comprehensively in real time, and with minimal equipment.

In this paper we propose a method to remove an occluder in a primary video, acquired from the user viewpoint, using pixels from a secondary video, acquired from a translated viewpoint. The secondary frame pixels are integrated seamlessly into the primary frame, with good continuity across the occluder contour. The result is a multiperspective frame, which shows most of the scene from the user viewpoint, except for the part hidden by the occluder, which is shown from the secondary viewpoint. The effect is a good approximation of the transparency effect needed to remove the occluder, which comes without the high cost 3D geometry acquisition.

We have tested our method with good results on a variety of challenging scenes, with intricate and dynamic geometry (see Fig. 1 and accompanying video). We have compared our method to ground truth and to state of the art occluder inpainting techniques, both qualitatively, through visual inspection, and quantitatively, through two traditional per-pixel similarity metrics and two perceptual image similarity metrics. The comparison reveals that our results are closer to ground truth than those of the prior art methods. Furthermore, since we only compute correspondences between frames along the occluder contour, and we do not engage in full 3D reconstruction, our method is fast, running at over 75Hz in a CPU implementation.

2 Prior Work

The removal of objects that hinder the user's viewing of a region of interest is a typical diminished reality problem [42]. By covering up a real object with the image of the background it occludes, one can make the object virtually invisible by creating a "see-through" effect [23]. The effect is implemented in three main steps: acquisition of the occluder background, modification of the acquired background to fit the occluder footprint, and compositing of the modified background into the user's view.

The background can be captured by the user in advance in the form of a dataset of images [25] or a detailed 3D model [29, 3]. When the hidden background objects are known, such as a specific person's face, a pre-captured dataset with angle-dependent images is sufficient [32]. An internet photo collection can also be used to delete a person in a video sequence, especially when the scene is a frequently photographed sightseeing spot [19]. Another approach is to search for the best matching disoccluded view of the target in an earlier frame [37, 12]. For a moving camera, SLAM was used to reconstruct the background progressively [27, 33]. These methods fail when the current background configuration has not been observed in any of the earlier frames. Both the pre-acquisition and the temporal resampling methods cannot handle highly dynamic scenes, such as a busy intersection.

A dynamic scene has to be acquired in parallel, from additional viewpoints. Using multiple cameras installed all around the scene, unstructured light fields can be acquired and used to render occluded rays [24]. Surveillance cameras have been used to see through walls [22, 17]. Multiple users, each with their own handheld camera, can capture the background for each other but the approach was only demonstrated for planar backgrounds with markers [10]. The background was also acquired with a remote-controlled robot equipped with a camera [4]. Like these prior methods, we acquire the background information with a secondary camera, but without background geometry assumptions or markers.

Once acquired, the background needs to be transferred to the primary camera viewpoint. By assuming the scene only consists of large planes, homography matrices have been used to warp the background to the user view [6]. Homographies have also been used to transfer the best matching background image from an internet image collection to the user view [19]. Another approach is to extract the 3D geometry of the background. Using stereo vision, the background

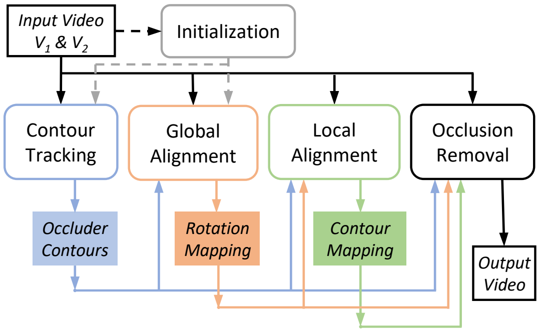

Fig. 2. System pipeline.

has been approximated with a set of small planes [42], or with a depth map [30]. The pixels missing due to disocclusion errors have to be filled in, e.g., through texture synthesis [9, 34]. The background geometry has also been acquired with the help of RGB-D cameras [21]. Reconstructing an accurate model of a complex 3D scene in real time remains challenging, and inaccurate geometry leads to output image artifacts, such as holes and tears. We bypass 3D geometry acquisition by computing a two stage bijective mapping that avoids such artifacts.

Inpaiting approaches approximate the occluded background from the surrounding, visible parts of the background [8]. Patched-based inpainting methods search iteratively in the neighborhood for best fitting patches [2, 5]. Prior information about the occluded part of the scene is useful for achieving a higher quality inpainting, such as of occluded human faces [16, 18]. Deep-learning has also been used to find matching patches at greater distances in the frame, with improved global consistency [20, 38, 15]. These methods have the advantage of not requiring a second video source, but they depend on the uniformity and predictability of the disoccluded background. As such, these methods are typically used to remove the annoying visual presence of the occluder, and not to provide a detailed and accurate description of the scene visible once the occluder is removed. Their goal is to erase the occluder well enough as to *not* draw the user's attention to the disoccluded part of the image; on the other hand, our method has the more challenging goal of allowing the user to focus precisely on the part of the scene that has been disoccluded.

3 Approach

We first give an overview of our occlusion removal pipeline.

3.1 Pipeline Overview

Given a primary input video stream and a secondary input video stream, acquired from a translated viewpoint, our method removes an occluder from the primary video using pixels from the secondary video, according to the pipeline shown in Fig. 2. First, an initialization stage defines the occluder contour in

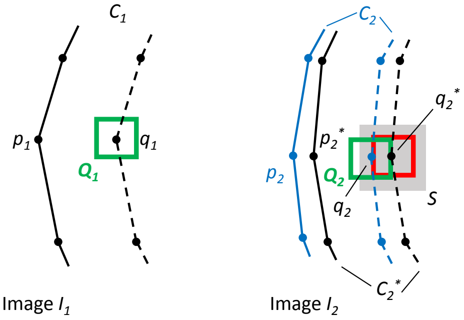

Fig. 3. Adjustment of approximate contour C_2^* to C_2 in image I_2, given the corresponding contour C_1 in image I_1 (Alg. 1). The algorithm searches for a better position for each inner contour vertex q_2^* over its neighborhood S; a good position q_2 yields a high color similarity between I_2 at q_2 and I_1 at q_1; q_1 is the inner contour vertex of C_1 corresponding to q_2^*. Once q_2^* is adjusted, the corresponding outer contour vertex p_2^* is adjusted to p_2 with the same offset.

the first frame of each video, and computes an approximate mapping between these two first frames. Then, pairs of primary and secondary video frames are processed in four stages: the contour of the occluder is updated in each of the two frames; an initial mapping between the pair of frames is computed as a rotation, by minimizing color differences outside the occluder; the initial mapping is locally refined at the occluder contour to enable splicing in the pixels from the secondary frame with good continuity to the surrounding primary frame pixels; finally, the occluder is removed from the primary frame by looking up its pixels in the secondary frame, using a concatenation of the global and local mappings.

3.2 Contour adjustment

Our pipeline relies several times on a contour adjustment algorithm, which we describe first. We define a *contour* as a pair of polylines that model the inner and outer boundaries of an object visible in an image. The inner contour is on the object and the outer contour is on the surrounding background. The inner and outer contours are needed to restrict color comparisons to the occluder or to its surrounding background. The two contours have the same number of 2D vertices, the segments of a contour do not intersect, each contour has disk topology, and contours do not have to be convex.

Our contour adjustment (Alg. 1) takes as input a first image I_1, a known contour C_1 in I_1, a second image I_2, and an estimate C_2^* of C_1 in I_2. The algorithm output is a contour C_2 in I_2 obtained by adjusting C_2^*. The algorithm adjusts C_2^* by moving one pair of vertices (p_2^*, q_2^*) at the same time, where p_2^* and q_2^* are corresponding vertices on the inner and outer contours (line 1). We describe the algorithm for the case when the adjustment proceeds along the inner contour (Fig. 3). Adjustment along the outer contour is similar.

Algorithm 1 Contour adjustment (also see Fig. 3)

Input: Image I_1, contour C_1 in I_1, image I_2, contour C_2^* in I_2
Output: Adjusted contour C_2

```
 1: for each vertex pair $(p_2^*, q_2^*)$ in $C_2^*$ do
 2:     $s_{max} = -\infty$
 3:     for each pixel center $q$ in neighborhood $S$ of $q_2^*$ do
 4:         $Q_1 = I_1$ patch centered at $q_1$
 5:         $Q_2 = I_2$ patch centered at $q$
 6:         $s_q = \text{sim}(Q_1, Q_2) + \lambda e^{-|q-q_2^*|^2/(2\sigma^2)}$
 7:         if $s_q > s_{max}$ then
 8:             $q_2 = q$, $s_{max} = s_q$
 9:         end if
10:     end for
11:     $p_2 = p_2^* + q_2 - q_2^*$
12: end for
13: RemoveSelfIntersections($C_2$)
```

The algorithm adjusts the position of q_2^* by searching its neighborhood S for a better location (lines 3–10). For each candidate location q, the algorithm computes color similarity between I_2 at q and I_1 at q_1. Color similarity is evaluated over square image patches Q_1 and Q_2 (lines 4–6). The inner contour vertex q_2 is adjusted every time image similarity improves (lines 7–8). Once the entire neighborhood of q_2^* has been searched, the contour vertex p_2 is adjusted by the same offset $q_2 - q_2^*$ as q_2^* (line 11). Once the inner and outer contours have been adjusted, C_2 is returned after any self intersection is removed (line 13). Our algorithm checks and removes self-intersections by traversing the outer contour; if two outer contour segments (p_2^i, p_2^{i+1}) and (p_2^j, p_2^{j+1}) intersect, where $i < j$, all outer contour vertices from p_2^{i+1} to p_2^j are removed, together with their corresponding inner contour vertices.

In line 6, the similarity between image patches Q_1 and Q_2 is computed differently based on whether images I_1 and I_2 are frames of the same video, i.e. both primary or both secondary, or not, i.e. one primary and one secondary. When I_1 and I_2 are from the same video, we compute similarity using negative sum of squared per-pixel color differences:

$$sim_{intra}(Q_1, Q_2) = -\sum_p (Q_1[p] - Q_2[p])^2. \tag{1}$$

When I_1 and I_2 are from different videos, we use a cosine similarity [31], in order to compensate for any large exposure and white balance differences between the two videos. Cosine similarity is computed by treating each patch as a vector, and by computing the cosine of the angle between the two vectors:

$$sim_{inter}(Q_1, Q_2) = \frac{\sum_p Q_1[p] \cdot Q_2[p]}{\sqrt{\sum_p Q_1[p]^2}\sqrt{\sum_p Q_2[p]^2}}. \tag{2}$$

Algorithm 2 Intra-frame video splicing for occlusion removal

Input: Primary video V_1, secondary video V_2
Output: Disoccluded primary video V_d

```
     // Initialization
 1:  $B_1^0 = \text{UserInputContour}(V_1^0)$
 2:  $C_1 = \text{RefineContour}(B_1^0)$
 3:  $C_2^* = \text{HomographyMapping}(C_1, V_1^0, V_2^0)$
 4:  $C_2 = \text{AdjustContour}(C_1, V_1^0, C_2^*, V_2^0)$
 5:  $R_1^0 = I$; $R_2^0 = \text{InitializeRotation}(C_1, V_1^0, V_2^0)$
 6:  for each frame $i$ do
         // Contour tracking
 7:      $C_1 = \text{AdjustContour}(C_1, V_1^{i-1}, C_1, V_1^i)$
 8:      $C_2 = \text{AdjustContour}(C_2, V_2^{i-1}, C_2, V_2^i)$
         // Global alignment
 9:      $j = k \times \lfloor i/k \rfloor$
10:      $R_1^i = R_1^j \times \text{RotationMapping}(C_1, V_1^j, V_1^i, R_1^{i-1})$
11:      $R_2^i = R_2^j \times \text{RotationMapping}(C_2, V_2^j, V_2^i, R_2^{i-1})$
12:      $R = (R_2^i)^{-1} \times R_1^i$
         // Local alignment
13:      $A_1 = \text{SalientContourPoints}(C_1)$
14:      $A_2 = \text{AdjustContour2}(A_1, V_1^i, R \times A_1, V_2^i, R)$
         // Occlusion removal
15:      $V_d^i = V_1^i$
16:      for each pixel $p \in C_1$ do
17:          $p' = \text{LookUp}(p, R, A_1, A_2, C_2)$
18:          $V_d^i[p] = \text{Blend}(V_1^i[p], V_2^i[p'])$
19:      end for
20:  end for
```

In addition to the color similarity value sim, the aggregate similarity score s_q (line 6) also includes a displacement term $\lambda e^{-|\delta p|^2/(2\sigma^2)}$, which favors small contour adjustments when sim values are similar. The displacement follows a normal distribution $\mathcal{N}(0, \sigma)$, where σ controls the distribution flatness. The weight λ achieves the right balance between the cosine similarity term and the displacement term. Both parameters σ and λ are set empirically in the 5-10 and 0.5-1.5 ranges, respectively, and tuned for each of the scenes. The displacement term aims to avoid large adjustments for marginal color similarity improvements, e.g. to avoid that a patch on an object edge slide up and down the edge without meaningful color similarity changes.

3.3 Video splicing for occlusion removal

Our pipeline implements Alg. 2, which takes as input the primary V_1 and secondary V_2 videos and removes an occluder from V_1 using pixels from V_2.

Initialization. The algorithm first performs a once per session initialization (lines 1–5). The user selects the occluder to be removed interactively by drawing

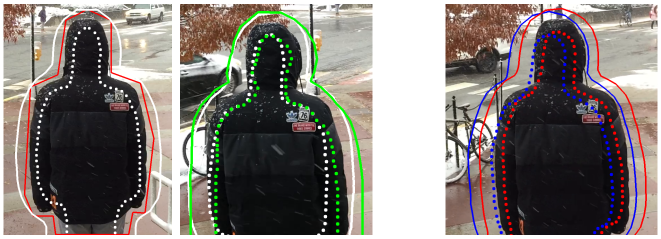

Fig. 4. Contour initialization (left and middle), and contour tracking (right). Left: first frame of primary video, with user drawn outer contour (red), and initial inner (white dots) and outer (white line) contours. Middle: first frame of secondary video, with contour lines transferred from primary video (white), and adjusted (green). Right: old contour in blue, adjusted contour in red.

in the first frame V_1^0 of the primary video an approximate piecewise linear outer boundary B_1^0 of the occluder (line 1, red line in Fig. 4 left). An approximate boundary that overestimates the occluder is sufficient, as the occluder will be removed with safety margins. Whereas other applications of segmentation have to recover an object contour with high-fidelity, as needed to paste it inconspicuously into a destination image, our application simply has to make sure that the entire occluder is discarded.

The outer boundary B_1^0 is refined to define the initial contour C_1 in frame V_1^0 (line 2), as follows: B_1^0 is rasterized to obtain a pixel mask M_1; M_1 is eroded to pixel mask M_1'; the inner contour of C_1 is defined as a subset of the outer pixels of M_1', i.e. pixels who have at least one of their eight neighbors not part of M_1' (white dots in Fig. 4 left); every inner contour vertex is moved outwards along its normal to define its outer contour vertex pair (white line in Fig. 4 left).

C_1 is used to initialize the occluder contour C_2 in the secondary video. C_1 is transferred to V_2^0 in two steps.

First, C_1 is taken almost all the way to its correct location in V_2^0 with a homography mapping from the C_1 region of V_1^0 to V_2^0; this provides an estimate C_2^* of the occluder contour in V_2^0 (line 3). The homography assumes that the occluder is a 3D plane, which is imaged by the two cameras with known intrinsic parameters. The homography is computed by detecting SURF features [7] inside the occluder region C_1 in V_1^0, and over the entire frame V_2^0. Each V_1^0 feature is matched to a V_2^0 feature with similar descriptor using FLANN [26]. The homography is determined by minimizing the reprojection error of corresponding features, using a RANSAC approach [11], which provides robustness to outlier feature correspondences. Fig. 4, middle, shows the outer and inner contours of C_2^* with white solid and dotted lines, respectively; C_2^* does not capture the occluder quite perfectly as the outer contour crosses into, and the inner contour crosses out of the occluder.

Second, C_2^* is adjusted to C_2, using our contour adjustment Alg. 1 (line 4). Since the frames provided to Alg. 1 belong to different videos, the cosine similarity metric is used. Fig. 4, middle, shows the adjusted contour C_2 with green solid and dotted lines.

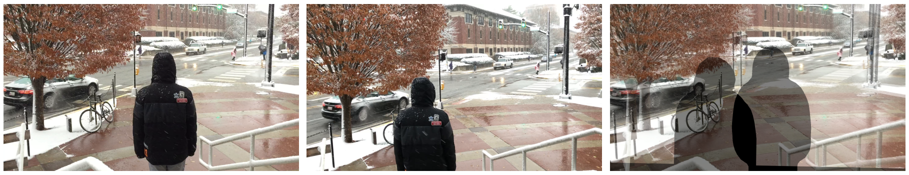

Fig. 5. Initialization of rotation from the first frame of the secondary video V_2^0 (left) to the first frame of the primary video V_1^0 (middle). The rotation is visualized by blending V_1^0 with the rotated V_2^0 (right). The rotation is recovered robustly (see alignment of distant parts of the scene), despite the considerable disparity between the two frames (see ghosting of near parts of the scene).

The algorithm maintains two arrays of 3D rotations, R_1 and R_2, one for each video. R_1^i rotates frame i of the primary video to frame V_1^0. R_2^i rotates frame i of the secondary video to V_1^0 as well, that the first frame of the primary video serves as a common reference. The last step of the initialization sets R_1^0 and R_2^0 (line 5). R_1^0 is the identity matrix. R_2^0 is computed by minimizing feature reprojection error, as described in Sec. 3.4. Fig. 5 illustrates R_2^0 by blending the rotated V_2^0 on top of V_1^0.

After initialization, each pair of primary and secondary video frames is processed with the four main stages of our pipeline.

Contour tracking. Contours C_1 and C_2 are updated in the current frames V_1^i and V_2^i, using the known contours in the previous frames V_1^{i-1} and V_2^{i-1} (lines 7–8). We use Alg. 1 again: the frame with the known contour is the previous frame, the frame where to adjust the contour is the current frame, and the estimate of the contour in the current frame is given by the contour in the previous frame. This no-motion contour prediction is sufficient because of the high frame rate of the videos compared to camera and occluder motion and velocity, and it bypasses the expense of computing a homography to provide the initial guess [13]. The frames are part of the same video, so similarity is computed using color difference. Fig. 4, right, shows the output of our contour tracking stage.

Global alignment. The algorithm has to compute a mapping from the primary video frame V_1^i to the secondary video frame V_2^i. For this, the algorithm first computes an approximate mapping. The approximate mapping is found by computing, for each video, the rotation of the current frame i to an earlier frame j *of that same video* (lines 9–11). Once the two rotations R_1^i and R_2^i are known, the approximate mapping R from V_1^i to V_2^i is easily obtained by leveraging the common reference V_1^0 of all rotations (line 12).

To find the rotation of the current frame i with respect to the earlier frames we use a more distant key frame j, and not the previous frame $i-1$, as consecutive frames would be too similar, and the alignment would drift. The key frames are spaced k frames apart (typically $k = 120$). Unlike for initial rotation computation (Sec. 3.4), here the rotation has to connect two video frames acquired by the *same* camera, from a *similar* viewpoint. Consequently, the global align-

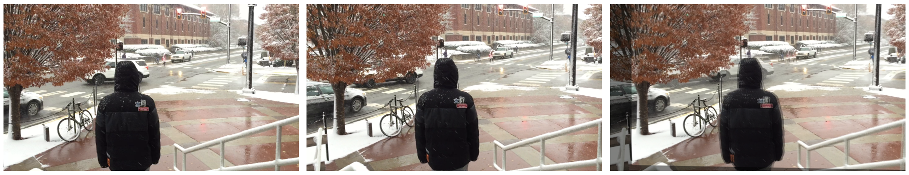

Fig. 6. Global alignment of two primary video frames (left and middle). The frames differ in view direction (see light post), and in time (see turning car). The blended visualization (right) shows that the global alignment recovers the rotation robustly (see alignment of distant scene), despite the motion in the scene (e.g., turning car) and the disparity of near objects (e.g., occluder and handrail).

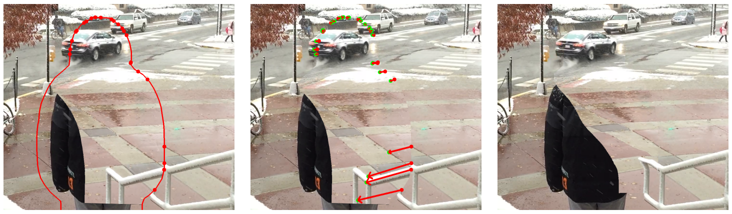

Fig. 7. Local mapping need (left), implementation (middle), and result (right). Left: disoccluding using only the global mapping results in discontinuities where near objects cross the occluder contour, e.g. where the sidewalk and handrail cross the red line in the left image. Middle: the local mapping connects primary frame salient contour points (red points) to their correspondence in the secondary frame (green points); the local alignment offset is larger for near objects. Right: disocclusion with continuity at occluder contour.

ment can be computed by directly minimizing color differences, bypassing the slower feature detection and matching. However, global alignment has to avoid the inconsistencies introduced by moving or near geometry, which create frame disparity even for small camera translations. Our global alignment computation is described in Sec. 3.5. Fig. 6 illustrates the accuracy and robustness of our global alignment stage.

Local alignment. The mapping R between frames V_1^i and V_2^i will be used to replace the occluder pixels in frame V_1^i with pixels from V_2^i. The mapping is approximate when the occluded scene is near and it has to be refined. The inaccuracy of the mapping is noticeable only at the occluder contour C_1, where the V_2^i pixels are spliced into V_1^i (Fig. 7, left). The algorithm computes a local alignment that alleviates color differences on each side of the occluder contour (lines 13–14). First, the outer contour of C_1 is sampled to gather a set of points A_1 with large color changes (line 13, and red dots in Fig. 7, middle). These points are better suited for computing the local alignment than the outer contour vertices because they do not sample wastefully regions of uniform color, and because they sample most regions with large color changes.

The newly defined outer contour A_1 is adjusted with an algorithm similar to Alg. 1, with two differences. The first difference is that the adjustment now proceeds following the outer contour, and not the inner one. Using Fig. 3 again, adjustment based on the outer contour is not concerned with the outer contour vertices and directly moves p'_2 to its better position p_2 that minimizes the color difference between I_2 at p_2 and I_1 at p_1. The second difference is that the adjustment now compares Q_1 to a rotated image patch Q_2, and not an axis aligned one (line 6 in Alg. 1). The rotated Q_2 is computed using rotation R. This more accurate comparison is now needed because the contour adjustment for the local alignment crosses between videos, and axis aligned patches do not match. Furthermore, adjustment is performed at the output frame cut line between the two video sources, so an inaccurate alignment would be readily visible. Fig. 7, middle, visualizes the displacement of the points of A_1 (red dots) to their correct locations A_2 (green dots). Fig. 7, right, shows the continuity achieved at contour boundary in the disoccluded frame using our local alignment.

Occlusion removal. Finally, the algorithm removes the occluder in the primary frame V_1^i (lines 15–19). The disoccluded frame V_d^i starts out as a copy of V_1^i (line 15), and then pixels p inside the contour are looked up in V_2^i. A pixel p is first rotated to p_r using R, and then p_r is offset with a weighted sum of offsets $a_2 - R \times a_1$, for all a_1 points in the vicinity of p (typically in a 45×45 neighborhood). We support several disocclusion visualization modes, such as cutaway, where p' completely replaces p, and transparency, where p and p' are blended together, with and without showing the contour of the occluder.

3.4 Rotation Initialization

The videos V_1 and V_2 are acquired from different viewpoints, so computing the rotation R_2^0 of frame V_2^0 to V_1^0 is challenging, as it does not benefit from frame to frame coherence. Indeed, the gap between V_1 and V_2 only has to be bridged for the first frame of V_2, as subsequent V_2^0 frames only have to be registered to their previous frame, whose rotation to V_1^0 is already known.

R_2^0 is computed by finding SURF features [7] in V_1^0, outside of C_1, and in V_2^0. V_1^0 features are matched to V_2^0 features using FLANN [26]. A pair of corresponding features is given a weight commensurate to the confidence in its correctness. The weight w_{ij} of a correspondence between a feature f_{1i} in frame V_1^0 and the most similar feature f_{2j} in frame V_2^0 is computed with:

$$w_{ij} = |f_{1i} - f_{2k}|/|f_{1i} - f_{2j}| \tag{3}$$

where f_{2k} is the feature second most similar to f_{1i}, and $|f_a - f_b|$ is the difference between the descriptors of two features f_a and f_b. The smallest possible weight is 1, when f_{1i} is equally similar to its best two matches, indicating the possibility of an ambiguous correspondence. When the second most similar feature f_{2k} is considerably less similar to f_{1i} than f_{2j} is, the correspondence is less likely to be incorrect, hence the larger weight. The reprojection error of corresponding features is minimized using a Gauss-Newton non-linear optimization [28], while also leveraging a RANSAC [11] approach to mitigate possible incorrect correspondences.

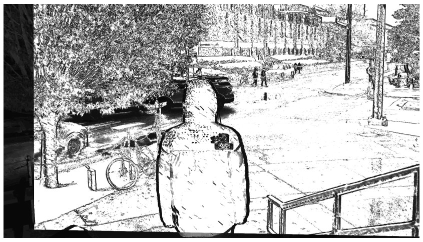

Fig. 8. Weights used in the global alignment from Fig. 6. Moving objects (e.g., car, pedestrians), and regions with high disparity (e.g., contour of near person), are assigned low weights, to reduce noise in the rotation computation.

3.5 Global Alignment

The global alignment computes the rotation of the current frame i to a previous key frame j, independently, for each of the two videos. We globally align two frames with a rotation because it provides a good approximation of the mapping between the frames without the prerequisite of scene geometry. We use the Gauss-Newton method [28] to find the three rotational degrees of freedom that minimize color difference.

Given a current frame V^i, a key frame V^j, the camera intrinsic matrix M (from a standard camera calibration process [41]), and a candidate rotation R from V^i to V^j, the color residual r_p at pixel p is:

$$r_p(R) = V^j[p] - V^i[M^{-1}(RM) \cdot p] \tag{4}$$

In Eq. 4, p is first unprojected from V_i, then rotated, and then projected to V_j. The stacked color residual vector over the entire frame is $\vec{r} = (r_1, \ldots, r_n)^T$, and the color error $E(R)$ is the ℓ_2 norm $|\vec{r}|$ of the residual vector. We use a left-compositional formulation. Starting with an initial estimate R^* given by the rotation of V_{i-1} to V_j, we compute an increment δR for each iteration:

$$\delta R = -(J^T J)^{-1} J^T \vec{r}(R), \text{ where } J = \frac{\partial \vec{r}(\epsilon \oplus R)}{\partial \epsilon}|_{\epsilon=0} \tag{5}$$

J is the derivative of the residual vector $\vec{r}$ with respect to an increment ϵ, and $J^T J$ is the Gauss-Newton approximation of the Hessian matrix of E. We then update the rotation estimate by multiplying it with the iteration's increment:

$$R = \delta R \oplus R \tag{6}$$

To gain robustness with outliers caused by moving objects, by the disparity of near objects, and by view dependent effects (e.g. reflections), the minimization is done in an iteratively reweighted fashion [14]. The weight of a pixel p equals the inverse $1/r_p$ of its residual. The weight is capped to avoid infinite weights when a pixel residual is very small. Fig. 8 visualizes the pixel weights for the global alignment from Fig. 6. The weighted rotation increment is:

Table 1. Average per-frame running times for the stages of our pipeline [ms], and overall frame rate [fps].

Stage	Contour tracking	Global alignment	Local alignment	Occlusion removal	FPS
Snow	1.3	1.8	3.3	1.8	122
Terrace	1.1	1.8	4.8	5.5	76
Atrium	1.6	1.8	5.1	2.9	88
Clutter	1.9	1.8	4.5	4.6	79
Crossing	1.3	1.8	5.3	1.6	100
Intersection	1.4	1.8	3.4	2.1	114

$$\delta R = -(J^T W J)^{-1} J^T W \vec{r}(R), \text{ where } W = \mathrm{diag}(1/r_1, \ldots, 1/r_n) \tag{7}$$

For speed, we perform this color residual minimization with a coarse-to-fine approach, that works at different levels of the image resolution pyramid. We start from the coarsest level of 30 × 17, as our frames have a 16:9 aspect ratio, and we stop at four levels deeper, i.e. at 480 × 270. The minimization converges at each level in between 2 and 4 iterations.

4 Results and Discussion

We have tested our occlusion removal method on several scenes, including *Snow*, *Terrace*, *Atrium*, and *Clutter* (Fig. 1), *Crossing* (video), and *Intersection* (Fig. 11). All scenes were abundantly dynamic, except for the *Clutter* scene, which was stationary. Each scene was acquired with two videos, captured with separate handheld phone and tripod mounted tablet cameras, from different viewpoints, matching the scenario described in the paper; the *Clutter* scene was acquired with a single handheld camera that revolved around the occluder, and the later frames were used to disocclude the earlier frames. The input and output videos have a 1,920 × 1,080 resolution. Our method worked well with all scenes, alleviating occlusions by creating a convincing transparency effect. We first report the running time of our method (Sec. 4.1), we discuss the quality of the occlusion removal achieved by our method (Sec. 4.2), we compare our method to ground truth (Sec. 4.3), and we compare our method to state of the art occlusion removal method (Sec. 4.4).

4.1 Time

We ran our disocclusion method for each pair of videos on an Intel i5-7600k workstation with a 3.8GHz CPU clock. Our implementation uses only the CPU (and not the GPU). The videos were played back at the original frame rate (60fps for *Snow* and *Terrace* and 30fps for the other scenes), and our method comfortably processed the frames in real time, with no precomputation.

Table 1 gives the average times for each of the four stages of our pipeline, as well as the average frame rate, which is at least 75fps. Contour tracking performance depends on the number of contour vertices, global alignment performance depends on the number of pixels in the resolution pyramid level used, local alignment performance depends on the number of salient contour points, and occlusion removal depends on the number of pixels in the occluder footprint. For *Snow, Atrium, Crossing* and *Intersection*, the slowest stage is the local alignment stage, which evaluates color differences with rotated and not axis aligned patches (parameter R of line 14 in Alg. 2). In addition to the cost of the rotation itself, comparing color between a rotated patch and an axis aligned patch introduces a bilinear interpolation per color comparison. For *Terrace* and *Clutter*, the occluder footprint is larger than for the other scenes, and occlusion removal takes slightly longer than local alignment. In all cases, our performance is sufficient even for 60Hz videos.

4.2 Quality

Our method handles well a variety of scenes, replacing the occluder pixels with pixels from the secondary video, with good continuity, as can be seen in the figures throughout the paper and in the accompanying video. The limitations of our method are discussed in the next section. Our method relies on a weak connection between the primary and the secondary frames: the frames are connected by an approximate mapping inside the occluder contour, and by a more rigorous mapping along the occluder contour. The weaker connection is faster to compute than the per-pixel correspondences used in structure from motion. Moreover, the weaker connection has the advantage of avoiding disocclusion errors.

Our method bypasses the computational expense of finding depth based on the disparity between the two video feeds, yet it avoids disocclusion errors that would plague such a depth-based approach.

Even if both videos are replaced with perfect RGBD streams, disocclusion errors can occur when the occluder is removed and the primary viewpoint gains line of sight to a part of the scene not visible from the secondary viewpoint. In Fig. 9, the primary viewpoint is O_1 and the secondary viewpoint is O_2. The secondary frame samples the green object from the left until B, and then the blue object from C towards the right. The primary frame is affected by the occluder FG. The primary frame sees the green object from the left until A, then the occluder, and then the blue object from E to the right. A depth-based, 3D occluder removal method leverages the perfect depth available at each secondary frame pixel to project the secondary frame pixels to their correct location in the primary frame. However, since the secondary frame does not sample the green object between B and D, the 3D occluder removal method leaves a gap, i.e. a disocclusion error, between B and C.

Our occluder removal method replaces the occluder pixels FG in the primary frame with the secondary frame pixels from A to E. Our local alignment makes sure that the primary and secondary frames are aligned at A and E. Our method does not recreate the primary view for the disoccluded part of the scene. Instead,

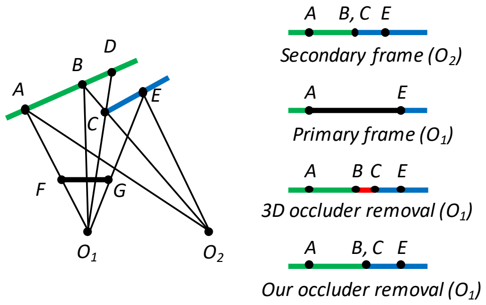

Fig. 9. Disocclusion error caused by 3D occluder removal. The secondary frame with viewpoint O_2 does not capture the green object between B and D. Even if the secondary frame has perfect depth per pixel, projecting the 3D samples of the secondary frame onto the primary frame will leave a gap between the projection of B and the projection of C. Our occluder removal method does not suffer from such a disocclusion error, as the mapping it uses does not allow B and C to separate in the primary frame.

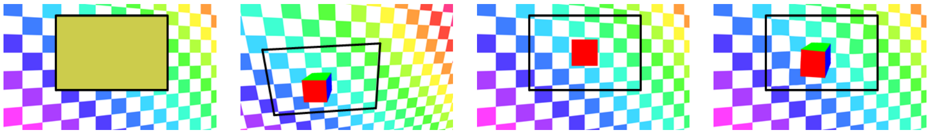

Fig. 10. Illustration of the multiperspective effect achieved by our disocclusion method: primary view, secondary view, ground truth transparency effect, and our output.

the pixels used to fill in the occluder shadow in the primary view come from the secondary view which has a different viewpoint, i.e. a different perspective on the disoccluded scene. This different perspective is maintained because the global alignment of our method is a homography and not a 3D warp. Our method does not remove the viewpoint difference but rather transitions from one viewpoint to another at the occluder contour.

Fig. 10 illustrates on a synthetic scene the multiperspective nature of the disocclusion effect achieved by our method. In the primary view (top left), the yellow rectangle occludes a cube with red, green, and blue faces. In the secondary view (top right), the cube is seen from a translated viewpoint, which reveals the blue and green faces. The first view occluder contour is shown with black in all images. on the background grid. Whereas the ground truth transparency effect only shows the front (red) face of the cube (bottom left), our visualization (bottom right) shows the cube from the secondary perspective, revealing its red, green, and blue faces. Our visualization changes perspective continuously based on the local alignment step which splices in the pixels from the secondary view.

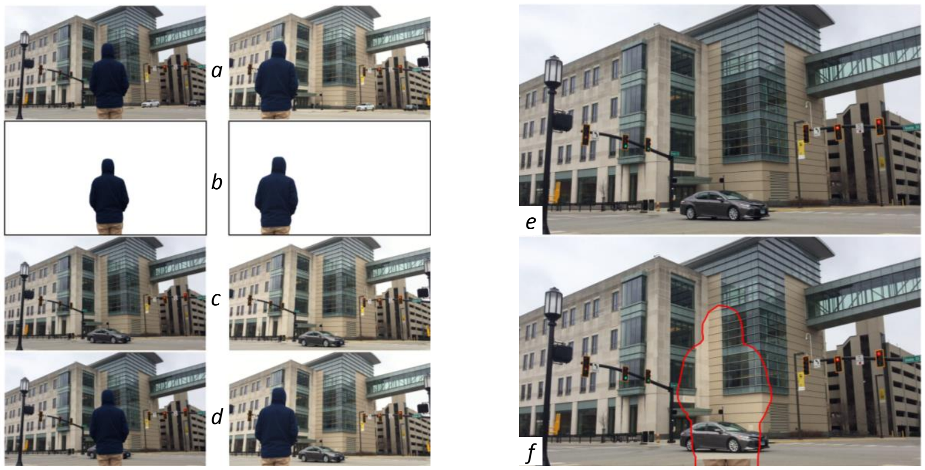

Fig. 11. Comparison of our disocclusion method to a ground truth transparency effect: (a) primary and secondary frames with occluder, (b) extracted occluder, (c) primary and secondary frames without occluder, (d) extracted occluder b inserted into frames c, (e) ground truth transparency effect, (f) output of our algorithm.

4.3 Comparison to ground truth

We have also compared our method to ground truth transparency over a real world video sequence (Fig. 11). For this, we recorded primary and secondary videos with the occluder obstructing both views (a), from which we extracted the occluder from each video (b); then we recorded primary and secondary videos without the occluder (c), which serve as ground truth; then we inserted the extracted occluders in each video (d), on which we run our algorithm. Our algorithm produces results (f) that are comparable to the ground truth (e). A sliver of the occluder remains at the bottom of our frame since the secondary view direction is tilted up and it does not cover that part of the occluder. Please also refer to the accompanying video.

4.4 Comparison to state of the art methods

We compare our results to those obtained with a commercial image processing platform, i.e. by using the *content-aware fill* tool of Adobe PhotoShop 2020 [1]. The tool is based on earlier occluder removal techniques [5, 36], updated based on image correction techniques based on deep-learning [38]. The tool relies on a second image from where to select pixels to inpaint the first image. In addition to the primary and secondary view frames, we provide as input the contour of the occluder in each frame. We also compare our method to a state of the art inpainting method based on partial convolution [20], and to two state of the art inpainting methods that use deep learning with typical convolution layers, namely *DeepFill v2* [39] and *Globally and Locally Consistent Image Completion*

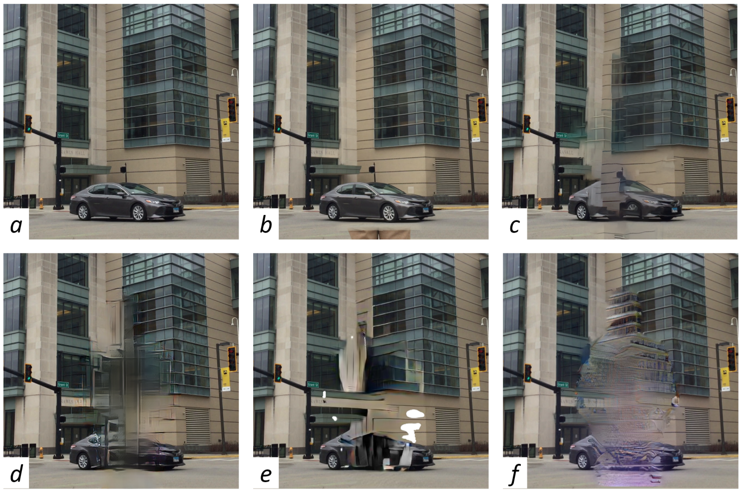

Fig. 12. Comparison to prior art methods: *a* ground truth, *b* our method, *c* content-aware fill [1], *d* partial convolution [20], *e* DeepFill [39], *f* GLCIC [15].

Table 2. Quality of occluder removal quantified by comparison to ground truth, for the images in Fig. 12, using two per-pixel comparison metrics, i.e. mean ℓ_1 error and PSNR, and two perceptual image comparison metrics, i.e. LPIPS, which measures image dissimilarity, so the lower the better, and SSIM, which measures image similarity, so the higher the better.

	ℓ_1 error (%)	PSNR	LPIPS	SSIM
Our method	**1.928**	**26.46**	**0.024**	**0.952**
Content-Aware Fill [1]	2.565	23.15	0.100	0.810
Partial Convolution [20]	3.476	21.61	0.143	0.818
DeepFill v2 [39]	4.046	20.41	0.170	0.810
GLCIC [15]	4.775	20.75	0.219	0.785

(GLCIC) [15]. All three inpainting methods find occluder replacement pixels in the input frame, and therefore they have the advantage that they do not require acquiring the scene from a secondary viewpoint.

Fig. 12 shows the results obtained by the five methods and the ground truth result, on the same frame. Our result shows significantly fewer artifacts and better continuity. We quantify the quality achieved by each method by comparing the result of each method to ground truth. We quantify the differences in terms of two per-pixel image comparison metrics, i.e. the mean ℓ_1 error and the peak signal-to-noise ratio (PSNR), as well as in terms of two perceptual image comparison metrics: the Learned Perceptual Image Patch Similarity (LPIPS) metric [40], and the Structural SIMilarity (SSIM) index [35]. Table 2 summarizes the comparisons which show that our result is closer to ground truth than that of the two other methods, even with the residual occluder sliver. Furthermore, the methods above were not designed for real-time video processing; according to our measurements, they require about one second of processing time per frame, on the same machine used to time our method. The authors of GLCIC report 0.5s on the GPU or 8.2s on the CPU for 1,024 × 1,024 images [15]. The authors of DeepFill v2 report 0.2s on the GPU or 1.5s on the CPU for 512 × 512 images [39]. We conclude that inpainting methods are better suited for the offline filling-in of small-area image holes, where they can extrapolate successfully from not too distant neighboring regions, and are less suited for filling-in in real time the large occluder footprint.

5 Conclusions. Limitations. Future Work

We have presented a method for removing an occluder from a video, by transferring pixels from a second video that captures what the first video should show if the occluder were not present. In actual use cases, the secondary viewpoint can be provided by a surveillance camera, or can be crowd-sourced, i.e., from the cameras of other people in the scene who volunteer to share their camera feed.

The pixels from the second video are spliced in with good continuity across the occluder contour. The method is based on the insight that a convincing transparency effect can be obtained without knowledge of 3D scene geometry. The method computes a mapping from the first video to the second video, which orients the second camera the same way as the first camera, but which does not attempt to translate the second camera viewpoint to the first viewpoint. The result is a multiperspective visualization, where the scene surrounding the occluder is shown from the first viewpoint, and the scene behind the occluder is shown from the second camera viewpoint. The two perspectives are connected seamlessly, with a local mapping that achieves a gradual transition from one viewpoint to the other.

The method achieves good results on a variety of scenes with intricate and dynamic geometry. We have shown that our method can produce results comparable to ground truth video obtained by recording the scene without occluder, and better results than prior occluder removal approaches. The method is fast, with a minimum frame rate of 75fps. Our method is fast because it only searches for correspondences between the two video frames along the occluder contour, and not for all the frame pixels, a fundamental advantage over traditional multi-camera computer vision pipelines. Our method is at a performance/quality point that prior art methods cannot approach, as shown in Fig. 12.

Our method does not require that any of the cameras be stationary — camera movement is accounted for by the global alignment stage. In fact, both cameras can move freely, as long as the region of interest is captured by the secondary camera. This first, rough alignment aligns well for the distant part of the scene, but it alone does not produce continuity across the contour (Fig. 7 middle). Continuity is achieved by our second step, i.e., local alignment along the contour, which is a key distinguishing feature with respect to prior methods.

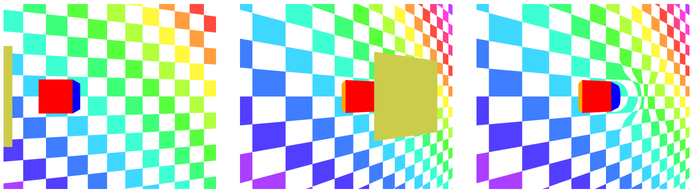

Fig. 13. Disocclusion of partially visible object: secondary view frame (left) primary view frame (middle) and our output (right).

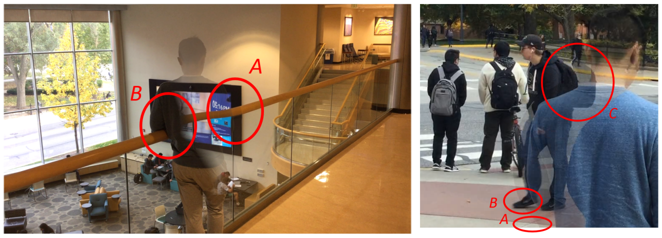

Fig. 14. Method limitation due to near object crossing the occluder contour: (left) perspective switch deformation (A) and extrapolation discontinuity (B); (right) the local mapping achieves continuity across the occluder contour for the pavement line (A), for the moving foot (B), but fails for the backpack (C).

Our method is interactive, allowing the user to specify the object to be removed by sketching a rough contour in the first frame, after which the contour is tracked automatically. Our pipeline is readily compatible with prior art techniques that can detect the "near" object from the first two frames of the primary and secondary feeds, which can be used to seed contour tracking automatically.

One limitation of our method pertains to near objects that cross the occluder contour. A near object is imaged from different directions by the two cameras, and therefore it has a different appearance in the two frames, a difference that cannot be alleviated by the global mapping rotation. This poses no problems when the near object is completely hidden behind the occluder (e.g. Fig. 10). However, when the object crosses the occluder contour, the object is distorted by the multiperspective effect, as it starts out in one perspective and ends in the other, the same way the subject is distorted in Picasso's cubist portraits that connect two views of the subject's face in a single painting. In Fig. 13, the switch from the primary to the secondary perspective occurs over the cube, distorting the cube. In Fig. 14, left, the handrail crosses the occluder contour in region A, where it switches from the primary perspective, outside the occluder, to the secondary perspective, inside the occluder. The switch is continuous, but the handrail is distorted as it is shown with two perspectives.

Another problem posed by partially occluded near objects arises when the secondary frame does not see everything the occluder hides in the primary frame. In such a case, a piece of the object is missing from both frames, and the local mapping cannot fill in the missing piece. In Fig. 14, left, the visualization appears discontinuous to a human observer in region B, who knows that the scene has one straight, uninterrupted handrail, and therefore expects that the disoccluded hand rail be aligned with the handrail reemerging to the left of the occluder. We call this an extrapolation discontinuity.

The perspective switch deformation and the extrapolation discontinuity problems are inherent to our method, in the sense that they occur even though our algorithms work as intended. The *Atrium* scene is a worst case scenario for these problems as the long, straight handrail makes them conspicuous. Future work could aim to reduce the perspective switch deformation by widening the area over which the switch between perspectives occurs; the extrapolation discontinuity could be reduced by leveraging or even pursuing a high-level understanding of the scene that maintains handrail continuity even though a handrail piece is missing from both frames.

A third problem posed by near objects that cross the occluder contour is that the local mapping fails occasionally (Fig. 14, right). For near objects, correcting the global mapping requires large offsets, and the search is less robust. The problem is exacerbated when the object moves quickly, and when the object does not have much texture, as is the case of the backpack and jacket in region C of Fig. 14, right. Using a small search neighborhood in the interest of performance reduces local mapping robustness. Future work could examine increasing the robustness of the local mapping with a strategy that leverages the image resolution pyramid to search over large distances to gain robustness without a significant performance trade-off.

Another limitation of the current implementation is that the visualization is not always perfectly stable. Presently, the set of salient points used by the local mapping is computed from scratch for every frame. Future implementations could limit the number of points replaced at every frame, in the interest of stability. Finally, the current implementation computes the global alignment with respect to a fairly recent key frame of the same video, and global alignment is computed across videos only once, for the first pair of frames. This works well for our sequences of 30s, but for longer sequences, global alignment drift could be a concern, which will have to be addressed by occasionally recomputing the global alignment between the current frames of the two videos.

We have shown that our method runs fast enough on a workstation, using only its CPU, to keep up with prerecorded videos. Future work should deploy our pipeline to phones and tablets, leveraging their GPUs. Future work could focus on absorbing into the local adjustment algorithm the latency of transmitting the secondary video to the user device where the disocclusion effect is computed. Another possible direction of future work is to use multiple secondary video streams to handle complex occlusions.

We describe a multiperspective framework for the continuous and non-redundant integration of multiple images, which, compared to traditional structure from motion, comes at the lower cost of only having to establish $O(w)$—and not $O(wh)$—correspondences between pairs of $w \times h$ images. This framework might find other applications, in augmented reality and beyond.

References

1. Adobe photoshop. https://www.adobe.com/products/photoshop.html, accessed: 2020-04-30
2. Álvarez, H., Arrieta, J., Oyarzun, D.: Towards a diminished reality system that preserves structures and works in real-time. In: VISIGRAPP (4: VISAPP). pp. 334–343 (2017)
3. Andre, E., Hlavacs, H.: Diminished reality based on 3d-scanning. In: Joint International Conference on Entertainment Computing and Serious Games. pp. 3–14. Springer (2019)
4. Avery, B., Piekarski, W., Thomas, B.H.: Visualizing occluded physical objects in unfamiliar outdoor augmented reality environments. In: Proceedings of the 2007 6th IEEE and ACM International Symposium on Mixed and Augmented Reality. pp. 1–2. IEEE Computer Society (2007)
5. Barnes, C., Shechtman, E., Goldman, D.B., Finkelstein, A.: The generalized patchmatch correspondence algorithm. In: European Conference on Computer Vision. pp. 29–43. Springer (2010)
6. Barnum, P., Sheikh, Y., Datta, A., Kanade, T.: Dynamic seethroughs: Synthesizing hidden views of moving objects. In: 2009 8th IEEE International Symposium on Mixed and Augmented Reality. pp. 111–114. IEEE (2009)
7. Bay, H., Tuytelaars, T., Van Gool, L.: Surf: Speeded up robust features. In: European conference on computer vision. pp. 404–417. Springer (2006)
8. Bugeau, A., Bertalmío, M., Caselles, V., Sapiro, G.: A comprehensive framework for image inpainting. IEEE Transactions on Image Processing **19**(10), 2634–2645 (2010)
9. Criminisi, A., Perez, P., Toyama, K.: Object removal by exemplar-based inpainting. In: 2003 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2003. Proceedings. vol. 2, pp. II–II. IEEE (2003)
10. Enomoto, A., Saito, H.: Diminished reality using multiple handheld cameras. In: Proc. ACCV. vol. 7, pp. 130–135 (2007)
11. Fischler, M.A., Bolles, R.C.: Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. Communications of the ACM **24**(6), 381–395 (1981)
12. Hasegawa, K., Saito, H.: Diminished reality for hiding a pedestrian using handheld camera. In: 2015 IEEE International Symposium on Mixed and Augmented Reality Workshops. pp. 47–52. IEEE (2015)
13. Herling, J., Broll, W.: High-quality real-time video inpaintingwith pixmix. IEEE Transactions on Visualization and Computer Graphics **20**(6), 866–879 (2014)
14. Holland, P.W., Welsch, R.E.: Robust regression using iteratively reweighted least-squares. Communications in Statistics-theory and Methods **6**(9), 813–827 (1977)
15. Iizuka, S., Simo-Serra, E., Ishikawa, H.: Globally and locally consistent image completion. ACM Transactions on Graphics (ToG) **36**(4), 1–14 (2017)
16. Jampour, M., Li, C., Yu, L.F., Zhou, K., Lin, S., Bischof, H.: Face inpainting based on high-level facial attributes. Computer vision and image understanding **161**, 29–41 (2017)
17. Kameda, Y., Takemasa, T., Ohta, Y.: Outdoor see-through vision utilizing surveillance cameras. In: Proceedings of the 3rd IEEE/ACM International Symposium on Mixed and Augmented Reality. pp. 151–160. IEEE Computer Society (2004)
18. Li, Z., Zhu, H., Cao, L., Jiao, L., Zhong, Y., Ma, A.: Face inpainting via nested generative adversarial networks. IEEE Access **7**, 155462–155471 (2019)

19. Li, Z., Wang, Y., Guo, J., Cheong, L.F., Zhou, S.Z.: Diminished reality using appearance and 3d geometry of internet photo collections. In: 2013 IEEE International Symposium on Mixed and Augmented Reality (ISMAR). pp. 11–19. IEEE (2013)
20. Liu, G., Reda, F.A., Shih, K.J., Wang, T.C., Tao, A., Catanzaro, B.: Image inpainting for irregular holes using partial convolutions. In: The European Conference on Computer Vision (ECCV) (2018)
21. Meerits, S., Saito, H.: Real-time diminished reality for dynamic scenes. In: 2015 IEEE International Symposium on Mixed and Augmented Reality Workshops. pp. 53–59. IEEE (2015)
22. Mei, C., Sommerlade, E., Sibley, G., Newman, P., Reid, I.: Hidden view synthesis using real-time visual slam for simplifying video surveillance analysis. In: 2011 IEEE International Conference on Robotics and Automation. pp. 4240–4245. IEEE (2011)
23. Mori, S., Ikeda, S., Saito, H.: A survey of diminished reality: Techniques for visually concealing, eliminating, and seeing through real objects. IPSJ Transactions on Computer Vision and Applications **9**(1), 1–14 (2017)
24. Mori, S., Maezawa, M., Ienaga, N., Saito, H.: Diminished hand: A diminished reality-based work area visualization. In: 2017 IEEE Virtual Reality (VR). pp. 443–444. IEEE (2017)
25. Mori, S., Shibata, F., Kimura, A., Tamura, H.: Efficient use of textured 3d model for pre-observation-based diminished reality. In: 2015 IEEE International Symposium on Mixed and Augmented Reality Workshops. pp. 32–39. IEEE (2015)
26. Muja, M., Lowe, D.G.: Fast approximate nearest neighbors with automatic algorithm configuration. VISAPP (1) **2**(331-340), 2 (2009)
27. Nakajima, Y., Mori, S., Saito, H.: 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). pp. 338–343. IEEE (2017)
28. Nocedal, J., Wright, S.: Numerical optimization. Springer Science & Business Media (2006)
29. Queguiner, G., Fradet, M., Rouhani, M.: Towards mobile diminished reality. In: 2018 IEEE International Symposium on Mixed and Augmented Reality Adjunct (ISMAR-Adjunct). pp. 226–231. IEEE (2018)
30. Rameau, F., Ha, H., Joo, K., Choi, J., Park, K., Kweon, I.S.: A real-time augmented reality system to see-through cars. IEEE transactions on visualization and computer graphics **22**(11), 2395–2404 (2016)
31. Singhal, A., et al.: Modern information retrieval: A brief overview. IEEE Data Eng. Bull. **24**(4), 35–43 (2001)
32. Takemura, M., Ohta, Y.: Diminishing head-mounted display for shared mixed reality. In: Proceedings of the 1st International Symposium on Mixed and Augmented Reality. p. 149. IEEE Computer Society (2002)
33. Tiefenbacher, P., Sirch, M., Rigoll, G.: Mono camera multi-view diminished reality. In: 2016 IEEE Winter Conference on Applications of Computer Vision (WACV). pp. 1–8. IEEE (2016)
34. Wang, L., Jin, H., Yang, R., Gong, M.: Stereoscopic inpainting: Joint color and depth completion from stereo images. In: 2008 IEEE Conference on Computer Vision and Pattern Recognition. pp. 1–8. IEEE (2008)
35. Wang, Z., Bovik, A.C., Sheikh, H.R., Simoncelli, E.P.: Image quality assessment: from error visibility to structural similarity. IEEE transactions on image processing **13**(4), 600–612 (2004)

36. Wexler, Y., Shechtman, E., Irani, M.: Space-time video completion. In: Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2004. CVPR 2004. vol. 1, pp. I–I. IEEE (2004)
37. Wu, M.L., Popescu, V.: Rgbd temporal resampling for real-time occlusion removal. In: Proceedings of the ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games. p. 7. ACM (2019)
38. Yu, J., Lin, Z., Yang, J., Shen, X., Lu, X., Huang, T.S.: Generative image inpainting with contextual attention. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 5505–5514 (2018)
39. Yu, J., Lin, Z., Yang, J., Shen, X., Lu, X., Huang, T.S.: Free-form image inpainting with gated convolution. In: Proceedings of the IEEE International Conference on Computer Vision. pp. 4471–4480 (2019)
40. Zhang, R., Isola, P., Efros, A.A., Shechtman, E., Wang, O.: The unreasonable effectiveness of deep features as a perceptual metric. In: CVPR (2018)
41. Zhang, Z.: A flexible new technique for camera calibration. IEEE Transactions on pattern analysis and machine intelligence **22**(11), 1330–1334 (2000)
42. Zokai, S., Esteve, J., Genc, Y., Navab, N.: Multiview paraperspective projection model for diminished reality. In: Proceedings of the 2nd IEEE/ACM International Symposium on Mixed and Augmented Reality. p. 217. IEEE Computer Society (2003)