RGBD Temporal Resampling for Real-Time Occlusion Removal

MENG-LIN WU, Purdue University
VOICU POPESCU, Purdue University

Occlusions disrupt the visualization of an object of interest, or target, in a real world scene. Video inpainting removes occlusions from a video stream by cutting out occluders and filling in with a plausible visualization of the object, but the approach is too slow for real-time performance. In this paper, we present a method for real-time occlusion removal in the visualization of a real world scene that is captured with an RGBD stream. Our pipeline segments the current RGBD frame to find the target and the occluders, searches for the best matching disoccluded view of the target in an earlier frame, computes a mapping between the target in the current frame and the target in the best matching frame, inpaints the missing pixels of the target in the current frame by resampling from the earlier frame, and visualizes the disoccluded target in the current frame. We demonstrate our method in the case of a walking human occluded by stationary or walking humans. Our method does not rely on a known 2D or 3D model of the target or of the occluders, and therefore it generalizes to other shapes. Our method runs at an interactive frame rate of 30fps.

CCS Concepts: • **Computing methodologies** → **Reconstruction**; **Image-based rendering**; • **Human-centered computing** → *Visualization.*

Additional Key Words and Phrases: Occlusion management, RGBD video, impostor

ACM Reference Format:
Meng-Lin Wu and Voicu Popescu. 2019. RGBD Temporal Resampling for Real-Time Occlusion Removal. 1, 1 (March 2019), 16 pages. https://doi.org/10.1145/3306131.3317025

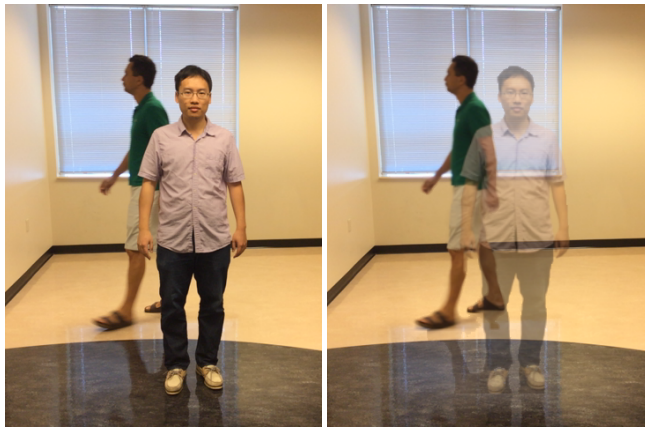
1 INTRODUCTION

Video streams are a widespread approach for visualizing real-world scenes in real time. However, video streams suffer from occlusion. In a dynamic scene, an object of interest, or target, can become temporarily hidden by other objects. In order to achieve an uninterrupted visualization of the target, one approach is to acquire the scene with multiple video streams, in the hope that the target is visible in at least one stream at any given time. Such a visualization with multiple streams implies a significant cognitive load for the user, who cannot monitor all streams in parallel, but has to examine the streams one at the time. Moreover, there is no guarantee that the target is visible in any one of the available streams.

Another approach is to rely on earlier frames to fill in the parts of the target that are occluded in the current frame. This inpainting approach for removing occlusions from video has the advantage of good visualization continuity, since the viewpoint does not change, and of simple acquisition, requiring only one video stream. Whereas there are real time video inpainting methods for removing occlusions of a static or of a rigid body target, prior art methods for deformable targets are too slow for real time performance.

In this paper we describe a novel method for video inpainting to remove occlusions of a deformable target in real time. Our method takes advantage of the fact that, in many cases of interest, such as when the target is a walking human, the target deformation is coherent, and the range of deformations is small. Our pipeline segments the partially occluded target from the current frame and inpaints the missing pixels by resampling from a matching disoccluded

Authors' addresses: Meng-Lin Wu, Purdue University, wu223@purdue.edu; Voicu Popescu, Purdue University, popescu@purdue.edu.



Video sequence A

Video sequence B

Fig. 1. Pairs of frames that illustrate our occlusion removal method: the original frame is shown to the left, and the frame output by our method is shown to the right. Our method runs at an interactive frame rate of 30 frames per second.

target from an earlier frame. Our pipeline takes as input a single video stream enhanced with a depth channel, i.e. an RGBD stream, acquired by a stationary computer tablet with a structured light depth camera accessory. The depth channel enables faster and more robust target segmentation and matching.

We demonstrate our method in the case of a walking human who is occluded by other stationary or walking humans (Fig. 1). Fig. 2 compares our output to truth using a synthetic occluder. Our method does not rely on a known deformable model of the targets and occluders, and therefore it supports general target and occluder shapes. For example, in Fig. 1 right, the target is a human pulling a wheeled bag. Our pipeline runs at interactive rates. We also refer the reader to the accompanying video that illustrates our method.

2 PRIOR WORK

The removal of image defects, and the subsequent completion of the image by inpainting, is a well-known problem since the invention of photography [Joyeux et al. 1999; Kokaram 2004]. Image defects such as scratches, dusts, text and

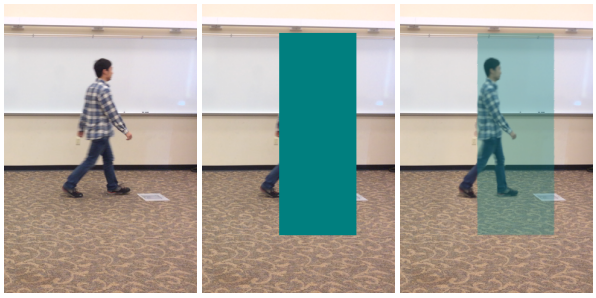


Fig. 2. Video sequence C — Comparison to ground truth: original frame (left), synthetic occluder added to original frame (middle), and frame without occlusion generated by our method (right).

scene clutter are cut out, and the remaining hole regions are inpainted using information taken from regions outside the hole, called source regions [Bertalmio et al. 2000]. Two fundamental approaches are low-level texture synthesis and and higher-level structure propagation.

Texture synthesis is concerned with computing a larger texture that is similar to a smaller initial sample texture. When large image defects such as unwanted objects are cut out, inpainting the large holes requires more than filling in with hole boundary colors. In these cases, a plausible background texture can be synthesized that is similar to surrounding source region texture, while maintaining continuity across the hole region boundaries [Bugeau et al. 2010; Efros and Leung 1999]. The plausibility of the inpainted image depends on the background being self-similar. Performance is improved by patch-based inpainting, which uses source region patches instead of pixels [Barnes et al. 2009; Barnes and Zhang 2017].

However, if cutting out the foreground reveals a background with salient geometry or non-repeating patterns, then the background structure must be extended into the hole region. The structure propagation approach extrapolates source region edges such as isophotes [Oliveira et al. 2001] or follows user provided guidance [Barnes et al. 2009; Sun et al. 2005] in order to inpaint plausibly. The structure propagation method is suited to inpainting cartoon-like image holes. Multi-level approaches that combine both texture synthesis and structure propagation are an active research area [Bertalmio et al. 2003; Criminisi et al. 2004; Elad et al. 2005].

A video is a linear sequence of images. To remove an object from a video segment, the object has to be cut out from multiple frames where it is visible, leaving behind a spatio-temporal volume defining the hole region to be inpainted. Video inpainting is more challenging than image inpainting because a) a large number of images have to be inpainted; b) the source region where to search for the missing colors is larger due to the addition of the time dimension; c) the inpainted holes must maintain temporal continuity across frames, in addition to maintaining spatial continuity across hole boundaries [Liu et al. 2009; Shiratori et al. 2006; Wexler et al. 2007].

Efficient video inpainting is possible, even in the case of dynamic camera poses, by propagating the mapping of inpainting pixels forward through successive video frames [Herling and Broll 2014; Kawai et al. 2016]. However, the background in the hole region has to be static and with simple geometry, such as a few planes. In the case of static foreground and background with significant depth separation, dynamic camera poses are leveraged to expose every part of the background through parallax. The static background can then be fully reconstructed and inpainted in every video frame [Xue et al. 2015]. Scene-space video inpainting removes the constraint on background simplicity by reconstructing complex scene geometry from depth images [Klose et al. 2015]. Although capable of filling in hole regions with complex background, the inpainting background scene points still need to be visible at other points in time

at the same scene-space position, so the static background restriction is not removed. Thus, both of these approaches are ill-suited for occlusion removal when the occluded background is typically dynamic and deformable.

For the more general problem of inpainting dynamic and deformable targets, such as walking humans, it is necessary to find the best source region, and to map the current hole region to the source region, for every video frame. The global optimization approach takes the entire spatio-temporal volume of the video sequence as search space for source regions [Barnes and Zhang 2017; Granados et al. 2012; Newson et al. 2014; Wexler et al. 2007]. By optimizing a visual quality energy function, the best mapping between the hole and source regions is obtained for inpainting the missing pixels. Although high visual quality is achieved, the performance is insufficient for interactive applications due to the large search space for source regions and the high cost to evaluate the energy function.

Object based methods improve performance by reducing the search space for source regions [Cheung et al. 2006; Huang and Tang 2016; Jia et al. 2005; Ling et al. 2011]. The target is segmented and tracked in the video. As occlusions occur, the hole region is inpainted with samples found in the target segment in video frames where the target is visible. Compared with the global optimization approach, the performance is improved when the target is small. However, to the best of our knowledge, video inpainting of deformable targets at interactive rates without user input has not yet been achieved.

3 SYSTEM OVERVIEW

Our system pipeline is shown in Fig. 3. The system takes as input a single RGBD video stream acquired from a fixed viewpoint showing a moving and deforming target that becomes occluded. The system outputs a video stream in which the target is disoccluded. The system processes each frame with a pipeline that has three main stages: segmentation, which identifies the occluded parts of the target, inpainting, which fills in the occluded parts of the target, and visualization, which shows the frame with the target unoccluded.

The frame is segmented both in 3D, by unprojecting pixels to a point cloud using the depth channel, and in 2D, using the color and depth channels. The output of the segmentation stage is an "incomplete billboard", which is a rectangular impostor in 3D that is texture mapped with the parts of the target visible in the current frame. The billboard texture has both color and depth channels. The incomplete billboard for the left pair in Fig. 1 is shown in Fig. 4. The segmentation stage is described in Section 4.

The incomplete billboard is inpainted using color and depth data from an earlier frame that shows the target in a similar pose. A matching earlier impostor billboard is found among the previously inpainted billboards, the current billboard is mapped to the earlier billboard using feature correspondences, and the occluded parts of the current billboard are filled in by resampling from the earlier billboard. The inpainting stage is described in Section 5.

The inpainted impostor billboard is used for an unoccluded visualization of the target. The occluder can be shown semi-transparently (Fig. 1, left), or it can be cut away (Fig. 1, right). The visualization stage is described in Section 6.

4 SEGMENTATION

The first step of occlusion removal is to identify the parts of the target that are occluded, which is done by segmenting the current frame (Alg. 1).

The algorithm for segmenting the current frame F_i takes as input a reference frame F_0 captured from the same view V_0 before the moving objects appear, and cylinders C_{i-1} bounding the moving objects that were computed at the previous frame. The algorithm proceeds in three major steps: 2D segmentation, 3D segmentation, and billboard

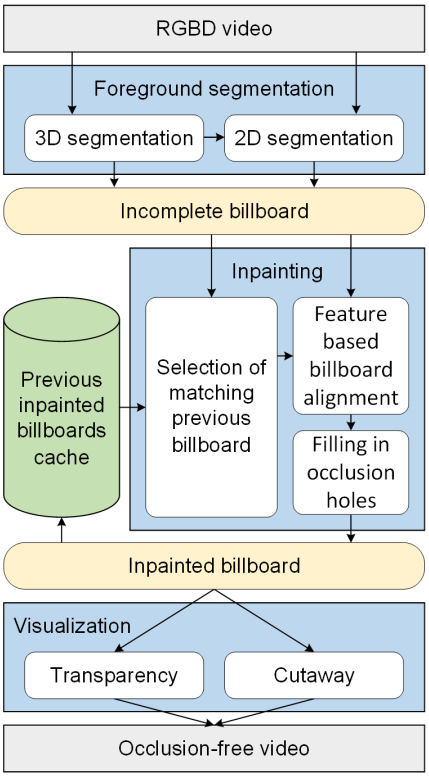


Fig. 3. System Overview

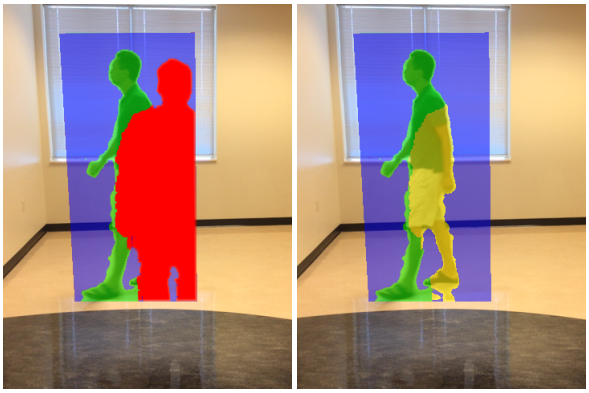


Fig. 4. Illustration of incomplete (left) and inpainted (right) billboard for the case shown in Fig. 1. The parts of the target visible in the current frame are highlighted in green, the parts of the billboard that are neither occluded nor part of the target are shown in blue, the parts of the billboard that are occluded are highlighted in red, and the parts of the billboard that are inpainted are highlighted in yellow.

Algorithm 1 Segmentation

```
1:  Input: current RGBD frame F_i, reference RGBD frame F_0, bounding cylinders C_{i-1} of moving objects in previous
    frame, camera view V_0.
2:  Output: billboards BB_i of current frame, bounding cylinders C_i of moving objects in current frame.
3:  // Step 1: 2D segmentation
4:  for each pixel (u, v) in F_i do
5:     FG(u, v) = F_i(u, v).z < F_0(u, v).z
6:  // Step 2: 3D segmentation
7:  for each pixel (u, v) in FG do
8:     if FG(u, v) then
9:        p = Unproject(u, v, F_i(u, v).z, V_0)
10:       Splat(p, I_top)
11: if target appears for the first time then
12:    BL = DetectBlobs(I_top)
13:    for all blobs BL_j in BL do
14:       C_ij = ComputeBoundingCylinder(BL_j)
15: else
16:    for all cylinders C_{i-1,j} in C_{i-1} do
17:       BL_j = ComputeBlob(I_top, C_{i-1,j})
18:       if Size(BL_j) > ε_B then
19:          C_ij = ComputeBoundingCylinder(BL_j)
20:       else
21:          C_ij = ExtrapolatePosition(C_{i-1,j})
22: // Step 3: Billboard construction
23: for all cylinders C_ij in C_i do
24:    BB_j.rectangle = VerticalCrossSection(C_ij)
25:    BB_j.texture = Project(F_i, V_0, BB_j.rectangle)
26:    for each pixel (u, v) on BB_j.texture do
27:       if F_i(u, v) == F_0(u, v) and !FG(u, v) then
28:          BB_j.texture(u, v).label = background
29:       else
30:          p = Unproject(u, v, Fi(u, v).z, V_0)
31:          if p is inside C_ij then
32:             BB_j.texture(u, v).label = visible
33:          else if p is behind C_ij then
34:             BB_j.texture(u, v).label = background
35:          else if p is in front of C_ij then
36:             BB_j.texture(u, v).label = occluded
```

construction. The 2D segmentation step (lines 3-5) partitions F_i into foreground and background by detecting changes in the depth channel with respect to F_0 (Fig. 5, top row).

The 3D segmentation step (lines 6-21) first computes a top view image I_{top} of the scene by unprojecting the foreground pixels to 3D points using the depth channel (line 9), and by splatting the 3D points into I_{top} using a camera with a downward view direction (line 10), see bottom left image in Fig. 5. Then, if F_i is the frame where the target first appears, the I_{top} is segmented into blobs using OpenCV's Simple Blob Detector algorithm [Bradski 2000], (line 12), and a vertical bounding cylinder is fitted to each blob (line 14). If the target was visible before (line 15), the blobs are tracked from frame to frame, instead of detected by searching blindly the entire top view image I_{top}. Each tracked blob BL_{ij} is found

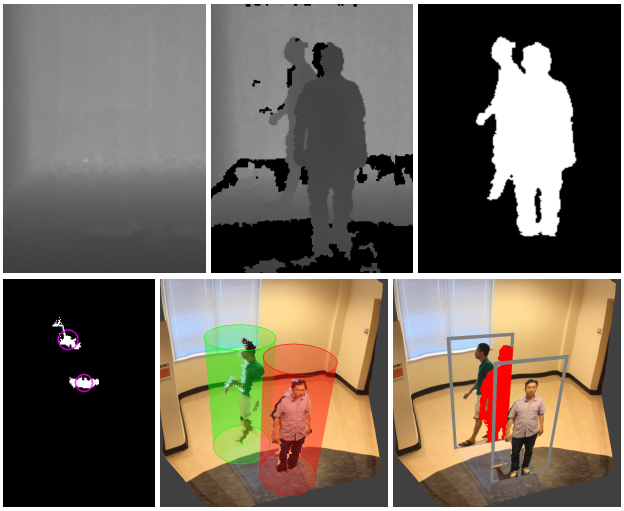


Fig. 5. Illustration of the steps of the segmentation stage. Top: depth channel of reference frame F_0 (left) and of current frame F_i (middle), and output of 2D segmentation (right). Bottom: top view image of foreground objects used for 3D segmentation (left), and visualization from a translated viewpoint of bounding cylinders (middle) and of billboards (right).

in I_{top} in the region where its bounding cylinder was found at the previous frame (line 17). If BL_{ij} is not present in I_{top} (line 18), the corresponding object is totally occluded and the its bounding cylinder C_{ij} is updated by extrapolating its position from the previous frames (line 21). We use a threshold ϵ_B below which the object is considered totally occluded. Furthermore, when the size of BL_J approaches the ϵ_B, the bounding cylinder is set by interpolating between the computed (line 19) and the extrapolated (line 21) cylinder (omitted for conciseness). The bounding cylinders are illustrated from a translated viewpoint in the middle image on the bottom row in Fig. 5.

The third step constructs an impostor billboard for each moving object based on its bounding cylinder. A billboard is a vertical rectangle defined by the cross section of the cylinder facing the camera (line 24), texture mapped with an RGBD image. The texture is computed by projectively texture mapping the current frame onto the rectangle (line 25), and then by labeling every texel as a visible part of the object (line 32, green in Fig. 4), as a visible part of the background (lines 28, 34, blue in Fig. 4), or as an occluded part of the object (line 36, red in Fig. 4). To label a texel, it is first compared to the reference frame F_0 (line 27). If the texel is distinct from the reference frame in the color channels or the depth value, then the labeling is decided based on the 3D segmentation of each object given by the bounding cylinder (lines 30-36).

5 INPAINTING

In the inpainting stage, the occluded pixels of the target billboard are further classified as background pixels, which are set to transparent, and as occluded target pixels, which are inpainted using samples from a previous target billboard, as described in Alg. 2.

The algorithm takes as input the target billboard BB_t computed by the segmentation stage at the current frame i, the cache of previously inpainted target billboards $IBBC$, and the index b_{i-1} of the frame where the matching billboard was found at the previous frame. The algorithm outputs the inpainted target billboard IBB_i and the index b_i where the matching billboard was found.

Algorithm 2 Inpainting

1: **Input**: target billboard BB_t of current frame i, cache of previous inpainted target billboards $IBBC$, frame index b_{i-1} of matching billboard from previous frame.
2: **Output**: inpainted target billboard IBB_i and frame index b_i of matching billboard at current frame.
3: // Step 1: Selection of matching billboard
4: $\epsilon_{min} = \infty$
5: **for** $j = \lfloor b_{i-1} - w/2 \rfloor$ **to** $\lfloor b_{i-1} + w/2 \rfloor$ **do**
6: $\quad \epsilon$ = MatchingError(BB_t, IBB_j)
7: $\quad$ **if** $\epsilon < \epsilon_{min}$ **then**
8: $\quad\quad \epsilon_{min} = \epsilon$
9: $\quad\quad b_i = j$
10: // Step 2: Feature based billboard alignment
11: $BB_t.features$ = FeatureExtraction($BB_t.texture$)
12: FeatureMatching($BB_t.features, IBB_j.features$)
13: Divide BB_t into square grid G
14: **for each** feature f in BB_t **do**
15: $\quad c$ = cell of G containing f
16: $\quad d = f - f.match$
17: $\quad$ **if** $||d|| < ||c.d||$ **then**
18: $\quad\quad c.f = f$
19: $\quad\quad c.d = d$
20: // Step 3: Filling in occlusion holes
21: **for each** missing pixel (u, v) in BB_t **do**
22: $\quad d = \sum_{c \in G} c.d\ \mathrm{K}(r, \sigma) / \sum_{c \in G} \mathrm{K}(r, \sigma)$, where $r = ||(u, v) - (c.f.u, c.f.v)||$
23: $\quad (u_j, v_j) = (u, v) + (d.u, d.v)$
24: $\quad BB_t(u, v) = IBB_j.(u_j, v_j)$
25: $\quad BB_t(u, v).label = IBB_j(u_j, v_j).label$
26: // Step 4: Update cache of inpainted billboards
27: $IBB_i = BB_t$
28: $IBB_i.features$ = FeatureExtraction($IBB_i.texture$)
29: $IBBC.Insert(IBB_i)$

The inpainting algorithm proceeds in three major steps: selection of matching billboard, feature based billboard alignment, and fill-in of occlusion holes.

To select the matching billboard (lines 3-9), the algorithm searches for the previous billboard that is most similar to the current billboard. A window of w billboards are considered, centered at the previously found best match. The width of the window w is empirically set at 0.3 times the distance, in number of frames, from the current billboard to the previously found best match. The index of the frame where the best matching billboard was found is recorded to center the search at the next frame (line 9). Given two billboards BB_t and IBB_j, the matching error is computed by minimizing the depth alignment error between the billboards as IBB_j is translated left-right and forward-backward with respect to BB_t. The depth alignment error for a left-right translation Δu and a forward-backward translation Δz is defined as the mean squared error between the depths z_t and z_j of the two billboards BB_t and IBB_j over the N overlapping pixels, as shown in Eq. 1.

$$\epsilon(\Delta u, \Delta z) = \sum_{u,v} (z_t(u, v) - z_j(u + \Delta u, v) + \Delta z)^2 / N \tag{1}$$

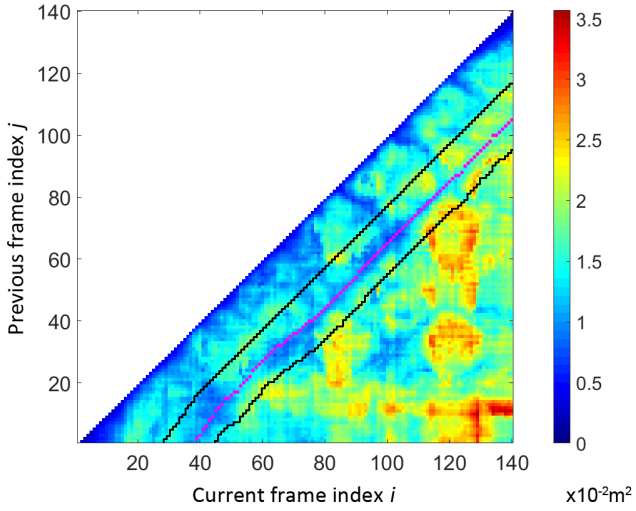


Fig. 6. The heat map visualization of the matching error. The x axis is the current frame index i, and the y axis is the previous frame index j, $j < i$. The search window is the region between the black lines, and the best matches found for each frame are marked by purple dots. The entire heat map is shown here to cover all possible pairs of (i, j), whereas only the (i, j) pairs inside the search window are calculated in Step 1 of Alg. 2.

Fig. 6 gives a heat map visualization of the matching error for the 140 frame sequence used in Fig. 1, left. A point (i, j) on the heat map corresponds to the matching error between a frame i and an earlier frame j. The black lines show the search window used by the algorithm, and the purple dots shows the best matches found for each frame. The black lines show that the search starts in the global minimum error valley. The plot shows that the period of motion is about 40 frames. The purple dots form a relatively straight line because the period does not change much, however, it is not constant. Our algorithm relies on the availability of a similar billboard, but it does not require that the motion be periodic.

If the target billboard computed by segmentation does not contain any occluded pixels, the inpainting algorithm stops. Step 1 has to be executed even for target billboards without occluded pixels in order to update the frame index b_i where to start the next matching billboard search.

Before actual inpainting can begin, a mapping between the current and previous billboard has to be computed. The rigid alignment computed at Step 1 during the selection of the matching billboard is not sufficient for quality inpainting. At Step 2, a non-rigid alignment is computed based on color features (lines 10-19). ORB features [Rublee et al. 2011] are extracted from the color texture of the billboard (line 11), and they are matched to the previously extracted features of the cached billboard IBB_j using the Hamming distance (line 12). The mapping between BB_t and IBB_j is computed efficiently with the help of a 2D grid G. The features of BB_t are assigned to cells in G (lines 14-19), which has a dimension of 20×40 cells, width×height. The grid G controls feature density in cells with great texture variation by culling all but one feature that are assigned to each cell. When multiple features are assigned to a cell, only the feature that is closest to its matched feature is kept (line 16-19).

Fig. 7 shows the billboard inpainted with and without non-rigid alignment. The left image shows extracted features in both the current and previous frames in colored circles. The matched features which survive culling are highlighted in cyan, while those that are discarded are colored red. The density of feature matches is effectively controlled by the

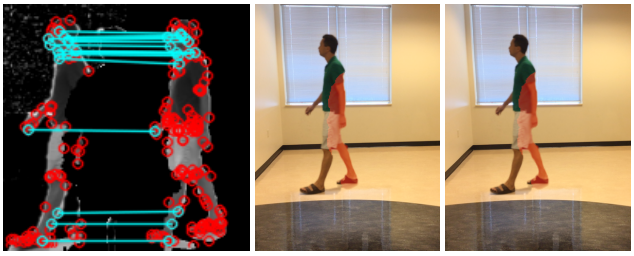


Fig. 7. Left: extracted ORB features. Matched feature pairs are highlighted in cyan. Middle: inpainted billboard without non-rigid alignment. Right: inpainted billboard with non-rigid alignment.

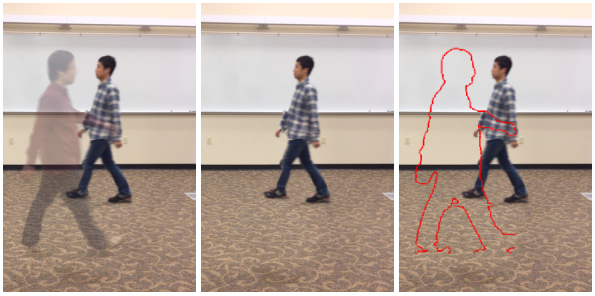


Fig. 8. Video sequence D — Occlusion-free visualizations: transparency (left), cutaway (middle) that completely removes the occluder, and cutaway with occluder silhouette (right).

culling process around the face and lower leg regions. The middle and right images show the inpainted billboards where the inpainted pixels are tinted red. With non-rigid alignment applied (right), the pixels around the shoulder are better aligned, while the pixels around the foot are unaffected and remain well aligned.

At Step 3, the grid with assigned features is used to compute the location from where to get color to fill in occlusion holes. For each occluded pixel of the current target billboard BB_t, the location in the best matching billboard IBB_j is given by a displacement d. The displacement is computed by filtering the displacements stored in the grid cells with a Gaussian kernel K centered at the current pixel (line 22). As a result, the displacement d is interpolated from displacements of nearby matched feature pairs, each weighted by the Gaussian kernel. The color and depth channels as well as the transparency label of the pixel are set using the corresponding pixel in the cached billboard (lines 23-25). If the corresponding pixel is transparent, i.e. it is a background pixel, the current pixel is also marked as transparent.

With all the missing pixels filled in, Step 4 of the algorithm extracts the complete set of color features from the inpainted target billboard. The billboard, along with extracted features, is inserted into the cache of inpainted billboards for use in future frames.

6 VISUALIZATION

Once the inpainting stage computes a complete impostor billboard of the target, the target is visualized occlusion free using standard occlusion-removal visualization techniques developed for synthetic scenes such as transparency and cutaway (Fig. 8). All objects are rendered on top of the reference frame efficiently as billboards with transparent texels. Correct visibility ordering of the billboards is readily available since the billboards are placed at the correct depth in 3D, leveraging the depth channel, as described above.

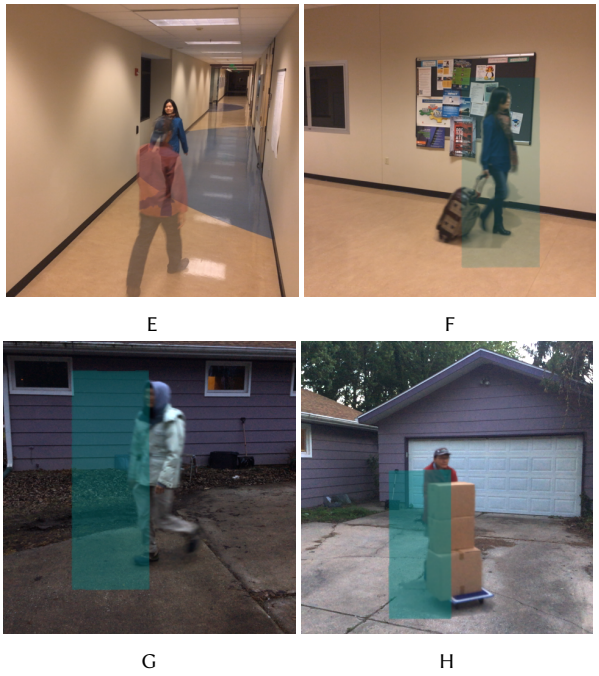


Fig. 9. Video sequences E to H — Occlusion-free frames generated with our method.

7 RESULTS AND DISCUSSION

We tested our system on several video sequences (Fig. 1, 2, 8, and 9) captured in both indoor and outdoor settings, with walking humans as targets and occluders. Some sequences have an eye-level viewpoint (e.g. sequences A and D) and some simulate a raised viewpoint like the ones used in surveillance applications (e.g. sequences B and E). In some sequences the target is a human rolling a wheeled bag or pushing a trolley cart, which illustrates that our method does not rely on a walking human model (e.g. sequences B, F, and H). In sequences C, F, G, and H the occluder is synthetic which provides ground truth for the occlusion free frames to which to compare the output of our method (Fig. 2).

The acquisition setup consists of a computer tablet (Apple iPad) with an on-board structured light depth camera accessory (Structure Sensor [Occipital 2013]). For our application, the tablet is set to acquire video at a resolution of 640 $\times$ 480 at 30Hz. The depth camera acquires depth at a resolution of 320 $\times$ 240 at 30Hz. The depth camera acquires depth frames in sync with the video frames. The tablet is mounted on a tripod to record RGBD videos from a fixed viewpoint.

7.1 Speed

We measured the performance of our pipeline by processing pre-recorded RGBD video sequences on an Intel Xeon E5-1660 3.3GHz workstation with 16GB of memory and with an NVIDIA Quadro K5000 4GB graphics card. The implementation mainly relies on the CPU, but uses OpenGL and GPU shaders written in GLSL for 2D segmentation and final composition. In all our tests, the frame rate is at least 27fps, with an average frame rate of 31.0fps (Table 1), or a frame time of around 32.2ms. Out of the three main stages of our pipeline, segmentation and billboard construction (Alg. 1) takes 6ms, and inpainting the target billboard (Alg. 2) takes 21ms. The visualization stage takes up the remaining frame time, i.e. 5ms.

Table 1. Frame rates achieved by our method [fps].

	Video sequence							
	A	B	C	D	E	F	G	H
min	27	28	29	29	28	28	30	29
max	35	38	33	38	37	34	35	37
average	30	31	31	32	31	30	30	33

Within the inpainting stage (Alg. 2), which only processes the target billboard and not the occluder billboards, steps 2 and 4 together take the most time at 13ms, or 40% of the total frame time. The cost arises from the FeatureExtraction function (Alg. 2, lines 11 and 28) which extracts ORB features for the purpose of non-rigid alignment between textures of current and previous target billboards. Performing feature extraction on downsampled billboard textures increases performance at the cost of alignment precision.

We achieve high performance by inpainting from a known region of a carefully selected previous frame. Compared to offline video inpainting methods, which may take image pixels or patches from disjoint regions across multiple frames, our method greatly reduces the search space for inpainting samples. Since our methods respects the inpainting source topology, the cost of the global optimization for the alignment of inpainting samples is also eliminated. Section 7.3 discusses the performance comparison in detail.

7.2 Quality

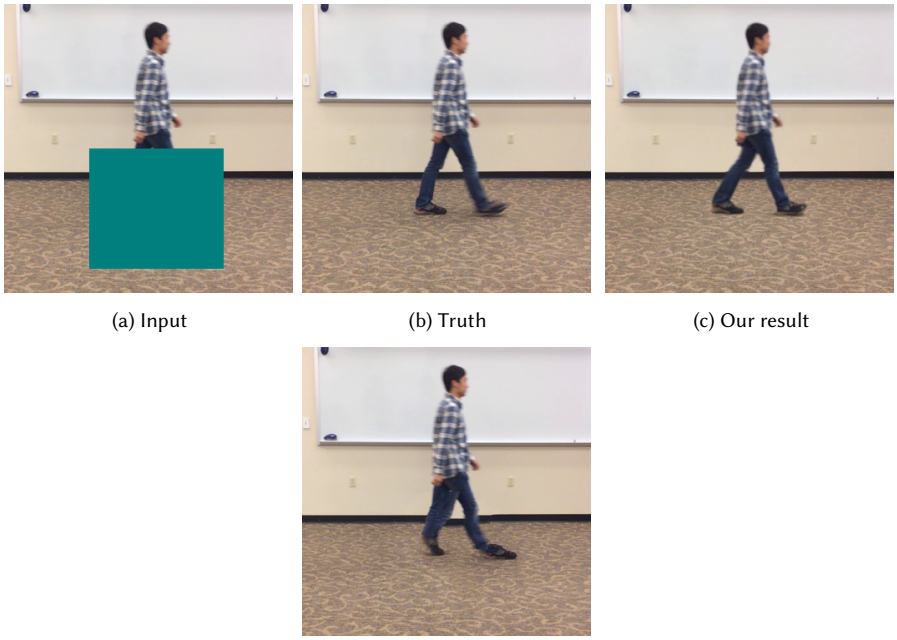
As shown in the images in this paper (Fig. 1, 2, 8, and 9) and in the accompanying video, our method produces quality occlusion-free visualizations of the target. Our method does not minimize per pixel error and rather focuses on providing a high-quality (blur free) visualization of the target in a similar pose to the one needed for the current frame. When the target is partially occluded, the inpainting conforms to the parts of the target that are visible. Our system supports a brief total occlusion of the target (Fig. 2), case in which the matching billboard is found by assuming a constant time offset to the frame used for inpainting (Alg. 2). Our method has good frame to frame coherence, providing a continuous, occlusion-free visualization of the target.

7.3 Comparison to prior work

We compare the result of our fast occlusion removal with a state-of-the-art patch-based video inpainting algorithm [Newson et al. 2014] (Fig. 10). To focus the comparison on the deformable and moving target object, we choose one of our video sequences with relatively simple background geometry. A dynamic artificial occluder, which occludes the target only partially, is added to the video sequence, while the original video, without the occluder, serves as ground truth reference. The prior work algorithm takes an additional binary occluder mask video as input.

The prior work method is able to inpaint the occluded parts of the target while maintaining edge continuity. It is also capable of inpainting both the flat and textured parts of the background. However, their method does not produce visually correct results for the target, as evident in the highly deformed legs. On the other hand, our method is able to inpaint the missing parts of the target by borrowing from an earlier frame when the missing parts were visible.

To compare the performance of the two methods algorithmically, we identify the two major operations common to our and prior work algorithms: search and reconstruction. The search operation in both algorithms searches within the video volume for best matching source pixels for inpainting the hole pixels. The reconstruction operation then



(a) Input (b) Truth (c) Our result

(d) Prior work

Fig. 10. Comparison between our method and prior work [Newson et al. 2014].

inpaints the hole by adapting source pixels to the hole region. For the purpose of algorithmic comparison, we denote the number of hole pixels H.

The search operation in our algorithm is efficient because the matching error is only computed in whole between the current billboard and the cached billboard (Alg. 2, Step 1). Furthermore, the matching error is computed only within the search window, w. The complexity is therefore $O(Hw)$. In comparison, the search operation in the prior work algorithm searches for a best matching patch to each hole patch, optimized over many iterations. The overall complexity is $O(k_l H n \mathcal{N}_p)$, where k_l is the number of iterations for optimization, n is the number of source patch candidates, and $\mathcal{N}_p$ is the patch size.

The reconstruction operation in our algorithm aligns matched pairs of features extracted from the current and source billboards (Alg. 2, Step 2 and 4). These sparse features are paired by brute force comparison. Overall, these steps incur a complexity of $O(k_e H + k_m f^2)$, where k_e and k_m are constants for feature extraction and matching. Step 3 executes in parallel on the GPU, and the cost is denoted by T_{GPU}. In the prior work algorithm, each hole pixel receives color from $\mathcal{N}_p$ source pixels, where each source pixel's weight is calculated by a patch-to-patch distance of $O(\mathcal{N}_p)$ complexity. The overall complexity is therefore $O(k_l H \mathcal{N}_p^2)$, where k_l is the number of iterations for optimization.

We estimate the constant factors for the complexity of the prior work algorithm with default parameters. Overall, the optimization iteration count k is on the order of 10^2. Furthermore, any pixel-to-pixel complexity is increased to patch-to-patch complexity, where a spatio-temporal patch defaults to a cube of 5^3 pixels. The optimization loop and patch size greatly scale up the run time of the prior work algorithm by a factor of 10^4.

We also compare the actual running times of the two methods. The prior work implementation is obtained from the authors, and it is executed with default parameters. The prior work implementation takes 3,053s to inpaint the 90

Table 2. Performance comparison

Algorithm	Search	Reconstruction
Ours (predicted)	$O(Hw)$	$O(k_e H + k_m f^2) + T_{GPU}$
Ours (measured)	1.7ms	13ms + 6.6ms
Newson (predicted)	$O(k_l H n \mathcal{N}_p)$	$O(k_l H \mathcal{N}_p^2)$
Newson (measured)	20s	3.1s

frame video segment for an average of 33.9s per frame. Our method takes an average of 32ms per frame for the same video segment, which is a significant performance improvement at 3 orders of magnitude. The prior work algorithm is implemented on the CPU, while certain steps of our pipeline (Alg.1, Step 1 and 3; Alg. 2, Step 3) run on the GPU. A detailed breakdown is provided in Table 2.

In order to objectively assess the reconstruction quality, we employ algorithmic metrics to compare the similarity between reconstructed images and truth images. We choose a learning-based metric which calculates the weighted cosine distance between feature vectors extracted from a convolutional encoder [Zhang et al. 2018]. Different from conventional pixel error metrics such as PSNR, learning-based metrics are able to infer high level image structure, which better evaluates inpainting of large regions of missing pixels. The implementation from the authors were used with default parameters for the AlexNet network [Krizhevsky et al. 2012]. Overall, our reconstructed images achieved an average distance of 0.010 ± 0.007 from truth images, significantly less than the prior method which achieved an average distance of 0.021 ± 0.007. Therefore, our method is shown to reconstruct images in the test sequence more similar to the ground truth than the prior method.

7.4 Limitations

Limitation in segmentation accuracy leads to artifacts in the impostor billboard textures in several ways: i) The 3D segmentation (Alg. 1, Step 2) can only segment objects separable by bounding cylinders. If the objects are intersecting each other, they cannot be separated cleanly. ii) Background subtraction relies on comparing the current pixel color to the known background color. However, dynamic objects cause lighting changes in the background. iii) The depth channel suffers from noisy or missing depth value, so pixels can be erroneously segmented. Our pipeline can easily leverage any advance in segmentation.

Our method assumes motion coherence over the occlusion time interval, so it does not support abrupt deviations from the motion pattern, such as, for example, the target stopping behind the occluder to look at their watch. Another limitation is the difficulty in predicting target motion when the target is heavily occluded for an extended amount of time. Our method does not rely on a constant motion period and it adapts the time offset to the past frame used for inpainting. However, when the target is occluded for a long time, this prediction of the position of the target becomes approximate. The difference between the predicted and actual target position can increase until the target reemerges, when the visualization snaps the target back to the correct position and pose.

Our method assumes that the target has the same appearance in the current frame as in the the inpainting source frame. This assumption is violated by a series of view-dependent effects, including parallax between the two frames as the position of the viewpoint relative to the target changes, and shadows and reflections of the target. It is also violated by non-uniform scene lighting which alters the target's appearance between the two frames.

8 CONCLUSIONS AND FUTURE WORK

We have implemented a system which removes occlusion to targets in RGBD video streams by target impostor inpainting. Our system operates at interactive rates by leveraging the depth information in several ways: depth keying augments color-based background subtraction; 3D point clouds constructed from depth images support fast object detection and tracking; billboards enhanced with per-texel depth enable efficient billboard to billboard correspondence computation. The system is demonstrated on a variety of real-world scenarios including one or more occluding objects, different camera perspectives, and different object geometries. Performance is greatly improved compared with prior work.

Our current work uses a single RGBD video stream and it shows the disoccluded video from the same reference viewpoint. Future work can explore the use of multiple input RGBD streams: multiple RGBD streams can acquire the target from different viewpoints over longer periods, constructing a comprehensive cache of the target's actions, which potentially increases the range of occlusions that our method can successfully inpaint. To leverage this comprehensive cache requires efficient cache indexing and searching. Inspired by the work of Liu et al. [Liu et al. 2017], we would like to investigate efficient search algorithms of best matching frame in the presence of large billboard caches accumulated from multiple video streams. A related possible improvement is to remove the requirement that the tablet be stationary during acquisition, where different approaches to video segmentation are called for.

Our method relies on a cache of previous target billboards, at least one of which needs to be similar to the current, partially occluded billboard. This is challenging for targets that exhibit a wide range of motions. For human targets, we demonstrated our method on various cases of walking motion while interacting with rigid objects. Our method does not require strict periodicity, and can handle acceleration and deceleration. This is an important case, as humans, animals, and birds move fairly uniformly, but with slight variations, over short distances. Objects that move rigidly like cars are much simpler cases.

Recent advances have been made in applying the convolutional neural network (CNN) to image and video inpainting problems [Liu et al. 2018; Wang et al. 2019]. These methods are able to inpaint complex images when the CNNs are trained with a dataset that is representative of the expected input. However, image inpainting using CNN does not straightforwardly preserve temporal coherence when applied to video processing [Liu et al. 2018], and fast video inpainting using CNN remains challenging due to the computationally costly 3D convolution over the input video volume [Wang et al. 2019]. We would like to explore potential hybrid approaches that leverage both the efficient billboard caching of our method and the generalized image reconstruction using CNNs.

REFERENCES

Connelly Barnes, Eli Shechtman, Adam Finkelstein, and Dan B Goldman. 2009. PatchMatch: A Randomized Correspondence Algorithm for Structural Image Editing. In *ACM SIGGRAPH 2009 Papers (SIGGRAPH '09)*. ACM, New York, NY, USA, Article 24, 11 pages. https://doi.org/10.1145/1576246.1531330

Connelly Barnes and Fang-Lue Zhang. 2017. A survey of the state-of-the-art in patch-based synthesis. *Computational Visual Media* 3, 1 (01 Mar 2017), 3–20. https://doi.org/10.1007/s41095-016-0064-2

Marcelo Bertalmio, Guillermo Sapiro, Vincent Caselles, and Coloma Ballester. 2000. Image Inpainting. In *Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '00)*. ACM Press/Addison-Wesley Publishing Co., New York, NY, USA, 417–424. https://doi.org/10.1145/344779.344972

M. Bertalmio, L. Vese, G. Sapiro, and S. Osher. 2003. Simultaneous structure and texture image inpainting. *IEEE Transactions on Image Processing* 12, 8 (Aug 2003), 882–889. https://doi.org/10.1109/TIP.2003.815261

G. Bradski. 2000. The OpenCV Library. *Dr. Dobb's Journal of Software Tools* (2000).

A. Bugeau, M. Bertalmio, V. Caselles, and G. Sapiro. 2010. A Comprehensive Framework for Image Inpainting. *IEEE Transactions on Image Processing* 19, 10 (Oct 2010), 2634–2645. https://doi.org/10.1109/TIP.2010.2049240

S. C. S. Cheung, J. Zhao, and M. V. Venkatesh. 2006. Efficient Object-Based Video Inpainting. In *2006 International Conference on Image Processing*. 705–708. https://doi.org/10.1109/ICIP.2006.312432

A. Criminisi, P. Perez, and K. Toyama. 2004. Region filling and object removal by exemplar-based image inpainting. *IEEE Transactions on Image Processing* 13, 9 (Sept 2004), 1200–1212. https://doi.org/10.1109/TIP.2004.833105

Alexei Efros and Thomas Leung. 1999. Texture Synthesis by Non-parametric Sampling. In *In International Conference on Computer Vision*. 1033–1038.

M. Elad, J.-L. Starck, P. Querre, and D.L. Donoho. 2005. Simultaneous cartoon and texture image inpainting using morphological component analysis (MCA). *Applied and Computational Harmonic Analysis* 19, 3 (2005), 340 – 358. https://doi.org/10.1016/j.acha.2005.03.005

Miguel Granados, James Tompkin, K Kim, Oliver Grau, Jan Kautz, and Christian Theobalt. 2012. How not to be seen—object removal from videos of crowded scenes. In *Computer Graphics Forum*, Vol. 31. Wiley Online Library, 219–228.

J. Herling and W. Broll. 2014. High-Quality Real-Time Video Inpainting with PixMix. *IEEE Transactions on Visualization and Computer Graphics* 20, 6 (June 2014), 866–879. https://doi.org/10.1109/TVCG.2014.2298016

J. Huang and X. Tang. 2016. A fast video inpainting algorithm based on state matching. In *2016 9th International Congress on Image and Signal Processing, BioMedical Engineering and Informatics (CISP-BMEI)*. 114–118. https://doi.org/10.1109/CISP-BMEI.2016.7852692

Yun-Tao Jia, Shi-Min Hu, and R. Ralph Martin. 2005. Video completion using tracking and fragment merging. *The Visual Computer* 21, 8 (2005), 601–610. https://doi.org/10.1007/s00371-005-0313-3

L. Joyeux, O. Buisson, B. Besserer, and S. Boukir. 1999. Detection and removal of line scratches in motion picture films. In *Computer Vision and Pattern Recognition, 1999. IEEE Computer Society Conference on.*, Vol. 1. 553 Vol. 1. https://doi.org/10.1109/CVPR.1999.786991

N. Kawai, T. Sato, and N. Yokoya. 2016. Diminished Reality Based on Image Inpainting Considering Background Geometry. *IEEE Transactions on Visualization and Computer Graphics* 22, 3 (March 2016), 1236–1247. https://doi.org/10.1109/TVCG.2015.2462368

Felix Klose, Oliver Wang, Jean-Charles Bazin, Marcus Magnor, and Alexander Sorkine-Hornung. 2015. Sampling Based Scene-space Video Processing. *ACM Trans. Graph.* 34, 4, Article 67 (July 2015), 11 pages. https://doi.org/10.1145/2766920

A. C. Kokaram. 2004. On missing data treatment for degraded video and film archives: a survey and a new Bayesian approach. *IEEE Transactions on Image Processing* 13, 3 (March 2004), 397–415. https://doi.org/10.1109/TIP.2004.823815

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. 2012. ImageNet Classification with Deep Convolutional Neural Networks. In *Advances in Neural Information Processing Systems 25*, F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger (Eds.). Curran Associates, Inc., 1097–1105. http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf

C. H. Ling, C. W. Lin, C. W. Su, Y. S. Chen, and H. Y. M. Liao. 2011. Virtual Contour Guided Video Object Inpainting Using Posture Mapping and Retrieval. *IEEE Transactions on Multimedia* 13, 2 (April 2011), 292–302. https://doi.org/10.1109/TMM.2010.2095000

Guilin Liu, Fitsum A. Reda, Kevin J. Shih, Ting-Chun Wang, Andrew Tao, and Bryan Catanzaro. 2018. Image Inpainting for Irregular Holes Using Partial Convolutions. In *The European Conference on Computer Vision (ECCV)*.

J. Liu, N. Akhtar, and A. Mian. 2017. Viewpoint Invariant RGB-D Human Action Recognition. In *2017 International Conference on Digital Image Computing: Techniques and Applications (DICTA)*. 1–8. https://doi.org/10.1109/DICTA.2017.8227505

Ming Liu, Shifeng Chen, Jianzhuang Liu, and Xiaoou Tang. 2009. Video Completion via Motion Guided Spatial-temporal Global Optimization. In *Proceedings of the 17th ACM International Conference on Multimedia (MM '09)*. ACM, New York, NY, USA, 537–540. https://doi.org/10.1145/1631272.1631350

Alasdair Newson, Andrés Almansa, Matthieu Fradet, Yann Gousseau, and Patrick Pérez. 2014. Video Inpainting of Complex Scenes. *SIAM Journal on Imaging Sciences* 7, 4 (2014), 1993–2019. https://doi.org/10.1137/140954933 arXiv:http://dx.doi.org/10.1137/140954933

Occipital. 2013. Structure Sensor. (2013). http://structure.io/

Manuel M Oliveira, Brian Bowen, Richard McKenna, and Yu-Sung Chang. 2001. Fast digital image inpainting. In *Proceedings of the International Conference on Visualization, Imaging and Image Processing (VIIP 2001), Marbella, Spain*. 106–107.

E. Rublee, V. Rabaud, K. Konolige, and G. Bradski. 2011. ORB: An efficient alternative to SIFT or SURF. In *2011 International Conference on Computer Vision*. 2564–2571. https://doi.org/10.1109/ICCV.2011.6126544

T. Shiratori, Y. Matsushita, Xiaoou Tang, and Sing Bing Kang. 2006. Video Completion by Motion Field Transfer. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, Vol. 1. 411–418. https://doi.org/10.1109/CVPR.2006.330

Jian Sun, Lu Yuan, Jiaya Jia, and Heung-Yeung Shum. 2005. Image Completion with Structure Propagation. *ACM Trans. Graph.* 24, 3 (July 2005), 861–868. https://doi.org/10.1145/1073204.1073274

Chuan Wang, Haibin Huang, Xiaoguang Han, and Jue Wang. 2019. Video Inpainting by Jointly Learning Temporal Structure and Spatial Details.. In *Proceedings of the Thirty-Third AAAI Conference on Artificial Intelligence*.

Y. Wexler, E. Shechtman, and M. Irani. 2007. Space-Time Completion of Video. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 29, 3 (March 2007), 463–476. https://doi.org/10.1109/TPAMI.2007.60

Tianfan Xue, Michael Rubinstein, Ce Liu, and William T. Freeman. 2015. A Computational Approach for Obstruction-Free Photography. *ACM Transactions on Graphics (Proc. SIGGRAPH)* 34, 4 (2015).

Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. 2018. The Unreasonable Effectiveness of Deep Features as a Perceptual Metric. In *CVPR*.