Multi-View Inpainting for Image-Based Scene Editing and Rendering

Theo Thonat[1], Eli Shechtman[2], Sylvain Paris[2], George Drettakis[1]
[1] Inria, [2] Adobe Research
{theo.thonat,george.drettakis}@inria.fr

Abstract

We propose a method to remove objects such as people and cars from multi-view urban image datasets, enabling free-viewpoint Image-Based Rendering (IBR) in the edited scenes. Our method combines information from multi-view 3D reconstruction with image inpainting techniques, by formulating the problem as an optimization of a global patch-based objective function. We use IBR techniques to reproject information from neighboring views, and 3D multi-view stereo reconstruction to perform multi-view coherent initialization for inpainting of pixels not filled by reprojection. Our algorithm performs multi-view consistent inpainting for color and 3D by blending reprojections with patch-based image inpainting. We run our algorithm on casually captured datasets, and Google Street View data, removing objects such as cars, people and pillars, showing that our approach produces results of sufficient quality for free-viewpoint IBR on "cleaned up" scenes, as well as IBR scene editing, such as limited displacement of real objects.

1. Introduction

Recent progress of IBR algorithms [26, 17, 6] allows free-viewpoint navigation in large regions of space. Combined with the massive data-acquistion efforts such as Google Street View or Microsoft Bing, IBR promises to provide the sense of "being there" for almost any location on the globe from within a web browser. However, a major downside of IBR is that it relies on multi-view photo datasets that must either be free of clutter (pedestrians, cars, signposts etc.) at capture time or requires painstaking editing to be usable for IBR. To edit a multi-view dataset for IBR, changes in both color and depth must be propagated to all views to keep the dataset consistent. Because every photo shows the scene from a different viewpoint, this propagation is challenging, especially when viewpoints are far apart from each other, i.e., in the wide-baseline case which is the focus of our work. Single-image inpainting, e.g., [7, 14], does not solve this problem because it does not ensure consistency between views. Neither does video inpainting, e.g., [23, 16], because it requires dense data, for instance to compute optical flow. Furthermore, inpainting for IBR must also infer consistent depth so that parallax can be properly rendered, which none of these techniques support.

In this paper, we propose a semi-automatic solution to multi-view inpainting and editing for IBR. Our algorithm takes a set of images and masks of content to remove, and inpaints image, normal and depth content coherent across views and consistent with the depth structure of the scene. With our algorithm, one can easily remove passers-by, cars, street signs, and other distractors that typically clutter IBR datasets, enabling the rendering of clean unobstructed views and even limited editing of the scene such as moving isolated objects.

To complete holes left in an image by a removed object, we use other views to "see" what is behind the removed object via IBR reprojection, or when such information is not available, e.g., a car big enough to hide a portion of the scene in all views, we use patch synthesis. Our first contribution is a unified approach to combine reprojected content with inpainting, so that consistent color, normals, and depth are produced across all views. We carefully balance these two sources of information so that inpainting progressively takes over reprojection when multi-view data are less reliable, e.g., coming from distant views or observed at grazing angle. Second, we introduce a multi-view patch search and multi-view consistent reconstruction method, taking into account the inaccuracies of the approximate 3D reconstruction, while exploiting the global 3D consistency it provides. We also propose a multi-view consistent initialization step which is an important element to the success of our approach.

Importantly, our inpainted multi-view datasets have color and depth consistent with the global 3D reconstruction, allowing the use of the result for IBR. We show two usage scenarios for our approach in Sec. 7.1 and

in the accompanying video. The first uses object recognition methods to remove classes of objects (e.g., cars, pedestrians etc.). In the second we provide a multi-view object removal interface to allow scene editing in a free-viewpoint IBR setting.

2. Previous Work

Single Image Inpainting. Criminisi et al. [7] proposed an inpainting algorithm which is able to retrieve basic structures, by using a well-chosen filling order, but works better with relatively small regions to inpaint. PatchMatch [3] finds approximate nearest neighbor matches between patches using random search. It can be used for inpainting and achieves a speedup of several orders of magnitude over previous work. Huang et al. [14] used planar information to guide the search space for patch matches, by estimating planar projection parameters and plane segmentation. In contrast, we leverage 3D information from multi-view reconstruction which provides an additional source of data for the patch search in our setting.

Video Completion. Wexler et al. [23] introduced a method for video sequence completion using space-time patches and a multi scale approach. Newson et al. [18] improved this technique by using an accelerated spatio-temporal search, and by introducing texture features to the patch distance to correctly inpaint video textures. Klose et al. [16] deal with a general sampling based algorithm for processing applications of a scene's video. The technique first collects a very large set of input videos pixels and then filters them iteratively before visually converging. Other methods such as [11] considers video inpainting as a labelling problem, but requires manual tracking of the object to inpaint. Multi-view information has been used to enhance low-resolution videos [4]. Video completion and enhancement methods provide important insight and can also be used as a methodological framework for multi-view datasets; the algorithms are nonetheless inherently different to ours since we assume wide-baseline photographs as input.

IBR and Multi-Image methods. Fitzgibbon et al. [9] use a patch-based approach for novel view synthesis in IBR. In contrast to our wide baseline data, they treat small-baseline datasets. In general IBR methods are designed to fill small holes due to depth disocclusions, and do not always adapt well to the more general inpainting problem we address.

Graph-cuts have been used when mixing images from different sources [1]; our approach is different in that we use multi-view reprojection and the associated *confidence* as a guide for patch-based inpainting. The shift-map algorithm [21] also uses graph-cut for hole filling, where the labels are image locations, while we will operate on color directly. Darabi et al. [8] extended the patch space search by adding rotated, re-scaled and photometrically transformed patches. Multiple images were used, but only as additional sources yielding good quality inpainting.

There has beeen some work using multiple views to remove objects from images. Whyte et al. [24] replace a user-defined target region from a query image using internet photographs of the same scene. Using homographies and photometric registrations, the method is able to blend the information from the entire dataset to synthesize encouraging results. Hays and Efros [12] use a large database of internet photos for image completion. The method is however inherently single-image and would not necessarily produce consistent results over a multi-view dataset. There has been plenty of work on RGB-D completion, including attempts to inpaint depth, typically restricted to stereo pairs [13]. In contrast, we target casual, wide-baseline capture, often with a mobile phone camera.

In recent work, developed concurrently with ours, Baek et al. [2] proposed a multi-view inpainting method jointly inpainting depth and color. This technique and ours share the same strategy of using depth and reprojected data to guide inpainting, but their scopes differ in major ways. Baek's technique is about image editing, and reconstructs per-image depth maps to handle occlusions, e.g., for inserting an object behind another one. Such depth maps are not sufficient for image-based rendering because they do not provide a consistent 3D representation shared across the images, which is needed for free viewpoint navigation. Our approach specifically addresses this scenario and generates such a global 3D representation.

3. Unified Multi-View Coherent Inpainting Algorithm

Our input is a multi-view set of images I_i of a scene, and a set of corresponding masks $\mathcal{M}_i$ which cover the parts of the images we want to remove. Masks are either automatically extracted *axis-aligned bounding rectangles* or regions created with a user-assisted process; we describe mask creation in Sect. 7.1. Our goal is to reconstruct images $\mathcal{I}_i$ in which the image content in the mask is removed and replaced by plausible content. We introduce a multi-view inpainting algorithm which builds on IBR and a patch-based algorithm [23], using PatchMatch [3] for search, guided by the multi-view data. A

Figure 1. (a) 3D reconstruction from multi-view input image dataset (b) Input images above, input masks below (c) Result of reprojection and mincut, remaining black pixels will be filled through patch-based inpainting (d) Inpainted result.

naive approach would be to first reproject as much data as possible from the other views, and use inpainting to fill in the remaining holes where data is missing. However, this simple strategy ignores that not all reprojected data are equally accurate. For instance, data observed at grazing angle are degraded because of foreshortening, and in practice, the 3D reconstruction and camera calibration are not perfect and data coming from distant views are less likely to be accurate.

3.1. Multi-View Input Data

Our typical input consists of 20–40 photos of a scene, with approximately 1.5–2m distance between shots. The images are then calibrated using Structure from Motion [25], and 3D reconstruction; we use CMPMVS [15]. This first step gives us an approximate mesh and calibrated cameras. Normals and 3D positions are provided approximately by the mesh, but regions not covered by the geometry remain. Several methods exist to propagate depth and/or normals in images; We extend the method of Chaurasia et al. [6] which propagates depth using oversegmentation to also propagate normals; these are precise enough to guide the patch matching process.

3.2. Problem Formulation

Considering an image I_i that we are inpainting, we obtain data from the other images I_j $(j \neq i)$ by reprojecting them into the mask $\mathcal{M}_i$ using the 3D reconstruction and camera calibration. In many cases, not all pixels in $\mathcal{M}_i$ are covered by the reprojected pixels; these correspond to the black pixels in Fig. 1(c) and Fig. 2(left). We write $\mathcal{M}_i^r$ the pixel coordinates in $\mathcal{M}_i$ which have a valid reprojection; $\mathcal{I}_r$ denotes the resulting image. We illustrate an example of $\mathcal{M}$, $\mathcal{M}_r$ and $\mathcal{I}_r$ in Fig. 2.

The pixels in $\mathcal{M}^r$ contain information from other views with varying accuracy. Where this information is reliable, we guide the patch synthesis to produce pixels similar to those in $\mathcal{M}^r$ by defining the following energy

Figure 2. Left: A crop of an input image showing the mask $\mathcal{M}$ as a dark region. Right: The resulting image $\mathcal{I}_r$ from reprojection into $\mathcal{M}$. The non-black pixels within the box are the region $\mathcal{M}^r$.

for an input image $\mathcal{I}_i$ with a mask $\mathcal{M}_i$:

$$E(\mathcal{I}_i|\mathcal{M}_i) = \sum_{p\in\mathcal{M}_i} \alpha(\mathcal{I}_r(p) - \mathcal{I}_i(p))^2 + (1-\alpha)\left(E_{\text{col}}(\mathbf{t}_p, \mathbf{s}_q) + E_{\text{3D}}(p)\right) \quad (1)$$

where $\mathbf{t}_p, \mathbf{s}_q$ are the coordinates of a target and a source patch respectively, centered at pixel p and q respectively. The term E_{col} is an extended patch difference measure which we explain in the following section, and E_{3D} imposes multi-view coherence (Sec. 6).

3.3. Algorithm

We minimize the energy in two steps: initialization, followed by iterative coarse-to-fine PatchMatch and voting. We alternate two EM steps for multi-view inpainting, PatchMatch and voting similar to Wexler et al. [23]

For the initialization, we first use 3D information to reproject other images into the current view (Sec. 4) and then perform a coarse initialization for the remaining unfilled pixels in a multi-view coherent manner (Sec. 5).

The first term is the squared distance of the reprojected image $\mathcal{I}_r$ with the inpainting results $\mathcal{I}_i$.

The second term is the patch difference measure [23]:

$$E_{\text{col}}(\mathbf{t}_j, \mathbf{s}_j) = D\left(W(\mathbf{t}_j), W(\mathbf{s}_j)\right) \quad (2)$$

where $W(\mathbf{t}_j)$ is the $N \times N$ patch centered at a pixel j and $W(\mathbf{s}_j)$ its associated nearest neighbor. The distance D is the sum of squared differences (SSD) of an eight-dimensional vector using the RGB values of $W(s_j)$ and

$W(t_j)$, the normals, and the gradient-based texture features of [18]. Normals and texture features are scaled using $\lambda_N = 0.5$, and $\lambda_{TF} = 0.75$ respectively.

We use several images of the multi-view dataset as sources for matching and present an algorithm to enforce multi-view consistency during initialization and voting. We encourage multi-view consistency through the term E_{3D}; (described in Sec. 6) and through a careful initialization which we will describe in Sec. 5.

Since the reprojection term and the multi-view patch synthesis terms are quadratic, the optimal solution is a linear blend of the patch synthesis result and I_r with alpha as the mixing coefficient. Specifically our algorithm blends the inpainted image $\mathcal{I}_i$ with the reprojected image $\mathcal{I}_r$ using the α weight for iteration t:

$$\mathcal{I}_i \leftarrow \alpha(\mathcal{I}_r) + (1-\alpha)\mathcal{I}_i \tag{3}$$

The new image $\mathcal{I}_i$ is then used in the next iteration of randomized PatchMatch, ensuring that the first term and the overall function $E(\mathcal{I}_i|\mathcal{M}_i)$ are minimized.

We summarize our approach in Algorithm 1. It is important to note that to achieve multi-view consistency all steps are done together for all the images being inpainted.

Algorithm 1: Global inpainting algorithm

Input: Multiview dataset with 3D reconstruction and binary masks
Result: Inpainted multiview dataset
for *each image* $\mathcal{I}_i$ *and mask* $\mathcal{M}_i$ *in the dataset* **do**
  for *each other image* $j \neq i$ *in the dataset* **do**
    Re-project into view $\mathcal{I}_i$ for pixels $\in \mathcal{M}_i$;
  $\mathcal{I}_i^r$ = min-cut over reprojections $\mathcal{M}_i$;
for *All images* $\mathcal{I}_i$ *in the dataset* **do**
  Initialize colors, normals, depth and Nearest-Neighbor Field (NNF) at coarsest scale ;
for *ScaleResolution* = *coarsest* **to** *finest* **do**
  repeat
    for *All images* $\mathcal{I}_i$ *in the dataset* **do**
      Reconstruct image $\mathcal{I}_i$ from NNFs with multi-view coherent voting;
      Blend image $\mathcal{I}_i$ with reprojection $\mathcal{I}_i^r$;
    for *All images* $\mathcal{I}_i$ *in the dataset* **do**
      Find NNFs with Patchmatch ;
  until *convergence*;
  if *ScaleResolution* $\neq$ *finest* **then**
    Upscale NNF ;
  else
    for *All images* $\mathcal{I}_i$ *in the dataset* **do**
      Reconstruct image $\mathcal{I}_i$ from NNFs and blend with reprojection $\mathcal{I}_i^r$;

4. Reprojection Initialization

We use image-based rendering (IBR) to reproject from different images of the input dataset to the target image $\mathcal{I}$. Methods using oversegmentation provide high-quality results [26, 1, 19]; however they are not specifically designed for *inpainting*. Such methods often assume that the missing data for a novel view is in the nearby input cameras, which is not always the case in our context. We adapt existing techniques [25, 22, 15, 19] by reprojecting several other input images that provide pixels in M_i, thus completing the empty region as much as possible (Fig. 2(right)). The quality of the reprojection degrades rapidly with distance between cameras, creating a trade-off between missing content and low quality reprojection depending on the number of images reprojected.

4.1. Reprojection with Mincut

Naive reconstruction of such reprojections (such as median or mean) does not provide satisfactory results, and IBR blending methods [5, 6] typically sacrifice quality for speed. We propose a solution based on a Markov Random Field (MRF) to choose between the input images, by considering – the pixels of – each reprojected image as a label, similar to [24].

We have a label ℓ for each possible source image (camera) and we seek a value ℓ_p for each pixel p. We first create median image of all reprojections, and following [24], we set unary $V_1(p, \ell_p)$ for a pixel p associated to a label ℓ_p as the squared difference to the median. We ignore pixels belonging to the mask for any input image, by setting $V_1(p, \ell_p) = +\infty$. Any target pixel with all labels equal $+\infty$ will be completed later by inpainting.

Images reprojected from other views do not always provide reliable information, notably when taken in directions far from the target view. We decrease the probability of a reprojected view with the angle between cameras as follows:

$$V_2(p, \ell_p) = \left(|\gamma_{\ell_p,\mathcal{I}}| + 1\right)^2 - 1 \tag{4}$$

where $\gamma_{\ell_p,j}$ is the angle between the cameras ℓ_p and j. The energy minimum is zero when the cameras align, and increases with the camera separation. We use a decreasing function of camera angle to allow almost linear falloff when approaching zero, and include a gradient

term [24], to preserve image structure:

$$W_1(p,q,\ell_p,\ell_q) = \|I_{\ell_p}(p) - I_{\ell_p}(q)\| + \|I_{\ell_q}(p) - I_{\ell_q}(q)\| + \lambda(\|\nabla I_{\ell_p}(p) - \nabla I_{\ell_p}(q)\| + \|\nabla I_{\ell_q}(p) - \nabla I_{\ell_q}(q)\|) \quad (5)$$

where $\nabla I_j(p)$ is the gradient in RGB space associated to pixel p in image j. The colors and gradient have bidirectional terms so they match in the two cameras ℓ_p and ℓ_q, and weight λ is a positive parameter, set to 10 in all our examples.

The global cost function we minimize with the MRF is thus (dropping dependence on ℓ_p for clarity):

$$E(L) = \sum_{p \in P} (V_1(p) + V_2(p)) + \sum_{(p,q) \in N} W_1(p,q) \quad (6)$$

where L is the labeling of pixels to inpaint P. The region is expanded by a few pixels using dilation to achieve a coherent visual transition between the target region and the rest of the image. N is the set of all pairs of neighboring pixels in P. The α value used in Eq. 1 is the residual energy value of the min-cut algorithm for each pixel, since it is based on the separation between cameras, which are a strong indication of high quality reprojection.

Color harmonization. Since the photographs come from a multi-view dataset, the same object observed from different viewpoints may appear with different colors, due to slight non-diffuse materials or subtle changes in lighting between shots. To avoid these artifacts, we use Poisson image compositing [20] after the mincut.

5. Coarse Initialization

Given the reprojected images $\mathcal{I}_r$, we need to initialize the color, depth and normal values for the remaining unfilled pixels, e.g., black pixels Fig. 2 (right). This step is critical for the overall success of the inpainting process. Sophisticated methods such as the Onion Peel approach [18] produce plausible results, but hinder multi-view coherence since they "push" the solution to different local minima for each input image. Instead, we interpolate information from the valid boundary pixels of the masks (i.e., containing information from the original image or reprojection) in a scanline fashion, which works well for the street-view scenarios we consider here; we discuss a possible generalization in Sec. 8.

In many cases, masks of different objects overlap in some views (e.g., a slanted view of a line of cars in a street). Interpolating over the entire merged mask tends to overblur the result. To avoid this, we introduce the notion of "object being removed", restricting the effect of interpolation to semantically similar regions. This is natural in our context, since masks are either automatically extracted, in which case they correspond to an object class ("car", "pedestrian" etc.) or are created by our user-assisted approach, in which case objects are typically being removed (see Sect. 7.1).

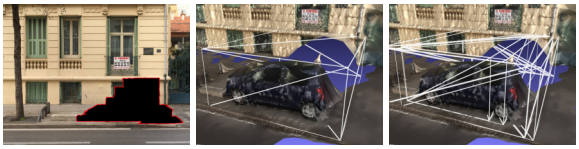

Figure 3. Left: 2D boundary of $\mathcal{M}_i^r$ in image i in red. Middle: Corresponding 3D bounding box. Right: Several bounding boxes combined together representing the car object.

To define objects, we take a 2D boundary of each 2D connected component of the mask in each image (shown in red Fig.3, left). We then create the 3D bounding box (white in the figure) of the corresponding 3D points contained within the 2D boundary (i.e., reprojected points that have depth). For each pair of 3D bounding boxes A and B we compute $w_{AB} = \max\left(\frac{V(A\cap B)}{V(A)}, \frac{V(A\cap B)}{V(B)}\right)$ and connect them if $w_{AB} > w_{\text{limit}}$ where $w_{\text{limit}} = 0.6$. These connected components are the objects subsequently used in initialization. For each scanline associated to the 2D boundary (Fig. 3(left)) we find the list of pixels on the left and right sides which have color, normals and depth available. For each scanline we linearly interpolate color, normal and 3D points between the two endpoints if their depths are available. If one endpoint does not have reliable depth, we propagate the color and normal of the other endpoint to the border of the image. Depth is considered reliable if it comes from the reconstructed mesh rather than the depth propagation step (Sec. 3.1). For 3D, we use the plane defined at the existing endpoint by its normal and propagate this across the scanline. To avoid excessive blurring, we introduce a heuristic to encourage vertical structures as detailed in the supplemental material.

Figure 4. Top view, $\mathcal{I}_r$ after reprojection. Lower row: intialization with multi view coherence. The reference view is in the middle. The different features are coherent across the views.

If we initialize each image separately, the result can

be very different in each view, whatever the initialization method used. To enforce multi-view coherence, we choose a reference view which will serve as a guide for all other views of the same object. For each object, we find a reference view by selecting the view with the largest number of scanlines with both left and right samples available, providing the largest amount of information. We reproject the lines of the reference view into all other views containing the object and for each pixel of a reference scanline. We copy the color and normal values into the target image and replace the initialization values. For depth, we use the correponding 3D point from the reference and reproject it into the current view. The effect of multi-view coherent initialization is shown in Fig. 4.

6. Multi-View Coherent Inpainting

Given multi-view coherent initialization, we can now proceed with our multi-view inpainting to minimize Eq. 1. Both the PatchMatch search step and the voting of Algorithm 1 will use multi-view information. It is important to balanace the tradeoff of imposing multi-view coherence, especially for IBR, and to avoid blurring which can happen to slight inaccuracies in the reprojection and the geometry used. We thus first need to define the neighbors of a given image to ensure multi-view coherence.

6.1. Defining Multi-View Coherent Neighbors

In street-side datasets, a given object is typically viewed by several input cameras. A simple nearest neighbor approach to define neighbors is insufficient since, if we consider a 2-neighborhood graph, disjoint graph cliques might be formed. Instead we use a minimum spanning tree approach on a graph connecting input views sharing objects with a low-weight edge or a path of low-weight edges.

Each node of the graph corresponds to an input view, with an edge between each pair of views. Consider the pair of views i and j; we intersect all bounding boxes of connected regions to be inpainted (see Sec. 5), and the weight w_{ij} is:

$$w_{ij} = \frac{d_{ij}}{\sum\limits_{(A,B)\in B_{ij}} \mathrm{V}\left(A \cap B\right)} \tag{7}$$

where d_{ij} is the distance between the two cameras i and j, and B_{ij} is the set of pairs of 3D bounding boxes induced by the camera i and j and V is the volume. The idea is to enforce the consistency between views that are close to each other and that share the maximum inpainted content.

We then find the minimum spanning tree of the graph. We traverse this tree from each node in order of smallest edge weight until we have K neighbors for each node. We use $K = 2$ for computational efficiency unless stated otherwise.

6.2. Multi-View Search and Coherent Voting

For a given image i, the search step to create the NNF in Algorithm 1 uses the image itself and its K neigbhors. The nearest neighbor of a patch is defined as the closest match amongst the matches in the three images.

Multi-view voting is expressed as the term E_{3D} in Eq. 1 and is given as follows:

$$E_{3D}(p) = \sum_{q\in S_p} ||I_{c_p}(p) - I_{c_q}(q)||^2 \tag{8}$$

where pixel p is from camera c_p and S_p is the set of pixels q from cameras c_q such that q belongs to the mask of c_q and reprojects into p. This reprojection is performed using the current depth estimation inside the hole.

We synthesize new 3D points which correspond to inpainted pixels with color in each view. The E_{3D} encourages the newly inpainted pixels corresponding to the same 3D point to have the same color.

We use the coarse 3D reconstruction and the inpainted depth to achieve multi-view coherence for voting. For a given pixel i from camera c_i, we look for patches that overlap this pixel and are centered at t_j, and their associated nearest neigbhor in the source s_j gives us a list of color candidates.

For multi-view consistency, we reproject the pixel in its neighboring views c_j. We then also look for patches in view c_j that overlap the reprojected pixel and we add the color associated to their nearest neighbor in the list of candidates. The final color is obtained by filtering all these candidates with the following weights:

$$w_j = e^{-\frac{||D(t_j,s_j)||^2}{2\sigma_i^2}} \times e^{-s||d_{c_i,c_j}||^2} \times Q(t_j). \tag{9}$$

Following [18], the first term favors source patches that are similar to their associated target patches. The parameter σ_i is defined at the 75th percentile of all distance $||D(t_i, s_j)||$. The second term gives more importance to closer views. The parameter s is the number of scales from the coarsest scale to the current scale. The idea is to reduce the influence of the multiview coherence as the upscaling occurs to avoid blurring at the finest scales. The quality term Q is a measure of how the nearest neighbor field is constant and in the same image. This is inspired by a similar approach previously used to improve single-image inpainting [8]. It is defined for a pixel target t_i as one plus the number of "high quality" neighbors t_j such

Figure 5. Above: Result without MV coherence; note missing blocks on the pavement for left image. Below: Result with MV coherence.

that $t_i - t_j = s_i - s_j$, where the neighors are taken from a 4-neighborhood around t_i.

Multi-view coherence has a significant effect on the results. As can be seen in Fig. 5, without coherence two different images of the dataset can have very different inpainted content. Using our approach, similar structures are created in the same position, which is central for good-quality IBR.

7. Results and Comparisons

We tested our results on seven multi-view datasets. Two are previously available [6] (e.g., Fig. 7 middle set), four were shot using a cellphone camera in a casual capture setting, (supplemental, Fig. 7 top, 8) and one is from a Google Street View data (Fig. 7 last dataset), which is extremely challenging since the baseline between panoramas is very large, causing SfM to often fail. Details of Google dataset processing are given in supplemental.

7.1. Usage Scenarios

We show two applications of our approach, one using a semi automatic method to remove specific classes of objects, the other introducing an interactive multi-view editor for IBR.

Semi-automatic Object Removal. We use the automatic object recognition approach of Gidaris et al. [10], for the classes "car", "people" and "motorbike", and we use the bounding boxes with the highest scores. This method works well in general; we increase the size of the bounding boxes to avoid missed regions and for some datasets we had to correct for manually missed regions (a few minutes per dataset).

These bounding boxes are used as masks for our multi-view inpainting. we use the inpainted images and the synthesized depth and run SLIC oversegmentation and additional depth synthesis [6] for remaining unreconstructed regions if required. We can now run IBR on the scene with the objects removed.

Editing Multi-View Captures for IBR. Another application of our approach is to not only remove objects in an IBR scene, but also to be able to move them. This enables – albeit limited – editing of multi-view captures with free-viewpoint IBR for the first time.

To do this, we built an interactive selection tool for multi-view datasets. We use an oversegmentation which allows selection of fine details to create good quality masks. The user can create variable width strokes on the object to select and segment it. We use multi-view stereo patches to reproject the strokes onto segments in the other views. The user can then cycle through views and complete or correct the segmentation, with little effort. We show an example of such a session in the accompanying video.

Figure 6. Left: input image. Right: novel view with the pillar extracted and moved, revealing content behind it. Our method allows such editing in a free-viewpoint IBR setting.

Once the object is segmented, we either use the resulting masks or combine them with masks from the automated approach. We extract the part of the image from the original image and render with a two-pass approach: first we render the background, and then render the extracted object, allowing us to edit the IBR scene, as seen in Fig. 6 and the video.

7.2. Comparisons

All comparisons were run by the authors of the previous papers. We compare with two single-image methods: PatchMatch (using content-aware fill in Adobe Photoshop) and the method of Huan et al. [14]. Single image methods have difficulty with large regions to inpaint, even when attempting to deduce information on planes as in [14]. We compare to two multi-view methods. The method of [24] produces good results in many regions, but is not multi-view consistent and is not designed for large regions with no reprojection. Even though the graphcut energy is infinity in masked regions, the method sometimes copies information from masked regions in other views or leaves the region black. The method of

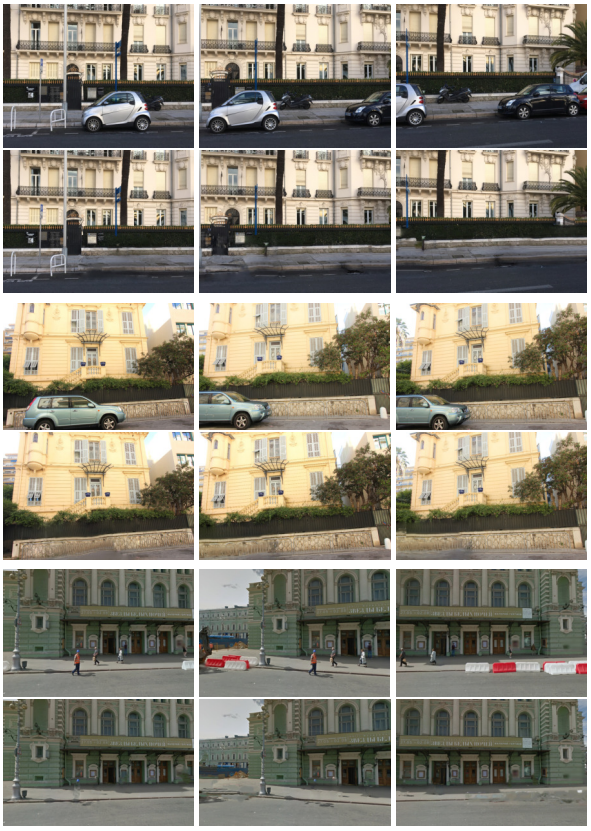

Figure 7. First 2 rows: results from one of our datasets. Middle 2 rows: the Yellowhouse datasets from [6]. Lower 2 rows: dataset reconstructed from Google StreetView. Input image above and and the resulting inpainting below for all cases.

[2] progressively builds multi-view information by successively visiting images. The results are of comparable visual quality to ours, but multi-view consistency is sometimes lost (Fig. 8). While depth information can be consistent across neighboring views (Fig. 9 (mid-right), it is not consistent at more distant viewpoints (left). The depth in this method is not globally consistent and lacks detail: e.g., the difference in depth between floor and wall in Fig. 9(left) is minimal. Given the lack of global depth consistency and detail, this solution cannot be used for IBR. While our synthesized depth is not perfect, it is sufficient for use with IBR as seen in the video (http://team.inria.fr/graphdeco/publications).

8. Conclusions and Discussion

Our results still suffer from some residual bluriness, that is hard to remove without breaking multi-view coherence, given the inaccuracies of the 3D reconstruction. Initialization could be improved using the 3D reconstruction to propagate structures in any direction, instead of the current horizontal interpolation on scanlines. Analyzing the 2D boundary of the reprojected region to identify directional structures could also improve results.

In conclusion, we demonstrated a multi-view consistent inpainting algorithm that for the first time enables editing of IBR scenes in a free-viewpoint setting.

Figure 8. Comparison with methods. We achieve much better image quality than all single-image inpainting techniques (rows 1-3). Compared to [2] which is multi-view, we see in the red boxes that our result has better multi-view consistency.

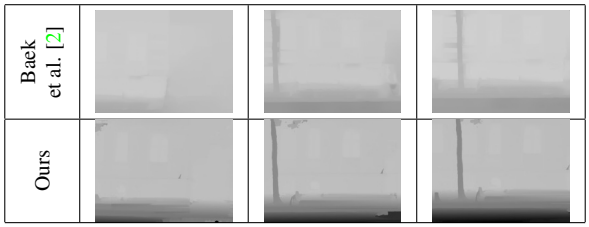

Figure 9. Comparison of depth of [2]; note incorrect depth on pavement (top left). Lack of global consistency and depth detail renders this method unsuitable for IBR.

9. Acknowledgments

Research funded by EU FP7 project 611089 CR-PLAY and French ANR project SEMAPOLIS (ANR-13-CORD-0003) and doctoral fellowship of the Région Provence-Alpes Côte d'Azur. We thank S. Bonopera for implementing the multi-view segmentation and IBR editing application.

References

[1] A. Agarwala, M. Dontcheva, M. Agrawala, S. Drucker, A. Colburn, B. Curless, D. Salesin, and M. Cohen. Interactive digital photomontage. 2004. 2, 4

[2] S.-H. Baek, I. Choi, and M. H. Kim. Multiview image completion with space structure propagation. In *Proc. IEEE Computer Vision and Pattern Recognition (CVPR 2016)*, Las Vegas, USA, 2016. IEEE. 2, 8

[3] C. Barnes, E. Shechtman, A. Finkelstein, and D. B. Goldman. PatchMatch: A randomized correspondence algorithm for structural image editing. *ACM Transactions on Graphics (Proc. SIGGRAPH)*, 2009. 2

[4] P. Bhat, C. L. Zitnick, N. Snavely, A. Agarwala, M. Agrawala, B. Curless, M. Cohen, and S. B. Kang. Using photographs to enhance videos of a static scene. In J. Kautz and S. Pattanaik, editors, *Rendering Techniques 2007 (Proceedings Eurographics Symposium on Rendering)*, pages 327–338. Eurographics, jun 2007. 2

[5] C. Buehler, M. Bosse, L. McMillan, and S. G. M. Cohen. Unstructured lumigraph rendering. In *SIGGRAPH*, 2001. 4

[6] G. Chaurasia, S. Duchene, O. Sorkine-Hornung, and G. Drettakis. Depth synthesis and local warps for plausible image-based navigation. *ACM Trans. Graph.*, 32(3):30:1–30:12, July 2013. 1, 3, 4, 7, 8

[7] A. Criminisi, P. Perez, and K. Toyama. Object removal by exemplar-based inpainting. 2003. 1, 2

[8] S. Darabi, E. Shechtman, C. Barnes, D. B. Goldman, and P. Sen. Image Melding: Combining inconsistent images using patch-based synthesis. *ACM Transactions on Graphics (TOG) (Proceedings of SIGGRAPH 2012)*, 31(4):82:1–82:10, 2012. 2, 6

[9] A. Fitzgibbon, Y. Wexler, and A. Zisserman. Image-based rendering using image-based priors. *International Journal of Computer Vision*, 2005. 2

[10] S. Gidaris and N. Komodakis. Object detection via a multi-region and semantic segmentation-aware cnn model. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1134–1142, 2015. 7

[11] M. Granados, K. I. Kim, J. Tompkin, J. Kautz, and C. Theobalt. Background inpainting for videos with dynamic objects and a free-moving camera. 2012. 2

[12] J. Hays and A. A. Efros. Scene completion using millions of photographs. *ACM Transactions on Graphics (TOG)*, 26(3):4, 2007. 2

[13] J. Howard, B. S. Morse, S. Cohen, and B. L. Price. Depth-based patch scaling for content-aware stereo image completion. In *WACV*, pages 9–16. IEEE Computer Society, 2014. 2

[14] J.-B. Huang, S. B. Kang, N. Ahuja, and J. Kopf. Image completion using planar structure guidance. *ACM Trans. Graph.*, 33(4):129:1–129:10, July 2014. 1, 2, 7, 8

[15] M. Jancosek and T. Pajdla. Multi-view reconstruction preserving weakly-supported surfaces. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pages 3121–3128. IEEE, 2011. 3, 4

[16] F. Klose, O. Wang, J.-C. Bazin, M. Magnor, and A. Sorkine-Hornung. Sampling based scene-space video processing. 2015. 1, 2

[17] C. Lipski, F. Klose, and M. Magnor. Correspondence and depth-image based rendering: a hybrid approach for free-viewpoint video. *IEEE T-CSVT*, 2014. 1

[18] A. Newson, A. Almansa, M. Fradet, Y. Gousseau, and P. Pérez. Video inpainting of complex scenes. *CoRR*, abs/1503.05528, 2015. 2, 4, 5, 6

[19] R. Ortiz-Cayon, A. Djelouah, and G. Drettakis. A bayesian approach for selective image-based rendering using superpixels. In *3D Vision (3DV), International Conference on*. IEEE, 2015. 4

[20] P. Pérez, M. Gangnet, and A. Blake. Poisson image editing. *ACM Trans. Graph.*, 2003. 5

[21] Y. Pritch, E. Kav-Venaki, and S. Peleg. Shift-map image editing. In *ICCV'09*, pages 151–158, Kyoto, Sept 2009. 2

[22] N. Snavely, S. M. Seitz, and R. Szeliski. Photo tourism: exploring photo collections in 3D. *ACM Trans. Graph.*, 2006. 4

[23] Y. Wexler, E. Shechtman, and M. Irani. Space-time completion of video. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 29(3):463–476, 2007. 1, 2, 3

[24] O. Whyte, J. Sivic, and A. Zisserman. Get out of my picture! internet-based inpainting. In *BMVC*, 2009. 2, 4, 5, 7, 8

[25] C. Wu, S. Agarwal, B. Curless, and S. M. Seitz. Multicore bundle adjustment. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pages 3057–3064, June 2011. 3, 4

[26] C. L. Zitnick and S. B. Kang. Stereo for image-based rendering using image over-segmentation. *International Journal of Computer Vision*, 75(1):49–65, 2007. 1, 4