Plane-Based Multi-View Inpainting for Image-Based Rendering in Large Scenes

Julien Philip
Inria, Université Côte d'Azur

George Drettakis
Inria, Université Côte d'Azur

Figure 1: Our multi-view inpainting method can remove objects such as cars (b) from all input images (a) of a multi-view dataset for Image-Based Rendering (IBR). This allows more flexible usage of IBR: by removing the cars, we avoid problems due to bad reconstruction which are more visible in novel views (c-d). Our method preserves perspective cues and provides clean separation between different planes (e.g., wall and sidewalk).

ABSTRACT

Image-Based Rendering (IBR) allows high-fidelity free-viewpoint navigation using only a set of photographs and 3D reconstruction as input. It is often necessary or convenient to remove objects from the captured scenes, allowing a form of scene editing for IBR. This requires multi-view inpainting of the input images. Previous methods suffer from several major limitations: they lack true multi-view coherence, resulting in artifacts such as blur, they do not preserve perspective during inpainting, provide inaccurate depth completion and can only handle scenes with a few tens of images. Our approach addresses these limitations by introducing a new multi-view method that performs inpainting in intermediate, locally common planes. Use of these planes results in correct perspective and multi-view coherence of inpainting results. For efficient treatment of large scenes, we present a fast planar region extraction method operating on small image clusters. We adapt the resolution of inpainting to that required in each input image of the multi-view dataset, and carefully handle image resampling between the input images and rectified planes. We show results on large indoors and outdoors environments.

CCS CONCEPTS

• **Computing methodologies** → **Rendering**; **Image-based rendering**; *Texturing*;

KEYWORDS

Inpainting, Multi-View, Image-Based Rendering

I3D '18, May 15–18, 2018, Montreal, QC, Canada

ACM ISBN 978-1-4503-5705-0/18/05…$15.00
https://doi.org/10.1145/3190834.3190846

ACM Reference Format:
Julien Philip and George Drettakis. 2018. Plane-Based Multi-View Inpainting for Image-Based Rendering in Large Scenes. In *I3D '18: I3D '18: Symposium on Interactive 3D Graphics and Games, May 15–18, 2018, Montreal, QC, Canada,* A. Bargteil, K. Mitchell (Ed.). ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3190834.3190846

1 INTRODUCTION

Recent Image-Based Rendering (IBR) solutions [Cayon et al. 2015; Hedman et al. 2016; Penner and Zhang 2017] provide high-quality free-viewpoint navigation, using only a *multi-view dataset* of photos of a 3D scene as input. However, the scene displayed is limited to the content in the input photographs. For 3D graphics applications, it is often necessary or convenient to remove objects from the scene, e.g., parked cars or furniture in a room. This can be achieved by *multi-view inpainting*, i.e., by removing the undesired objects in each input image and filling corresponding pixels and depth using inpainting or completion [Huang et al. 2014; Thonat et al. 2016]. Single-image inpainting methods [Bertalmio et al. 2000; Criminisi et al. 2004] are not designed to treat multi-view datasets even when they handle effects such as perspective [Huang et al. 2014]. Recent work provides initial solutions to this problem [Baek et al. 2016; Thonat et al. 2016], but suffers from four limitations: 1) multi-view coherence is applied progressively across neighboring images and is often inaccurate or incomplete, 2) perspective effects are not correctly reproduced during inpainting, resulting in visual artifacts and blurring, 3) the quality of depth synthesis is insufficient and 4) the methods are not designed to handle large datasets, since they often use expensive algorithmic solutions operating on all images in the dataset. We target scenes containing man-made structures, corresponding to city blocks or apartments, containing up to hundreds of input images.

The key to overcoming these limitations is to perform inpainting in *intermediate, locally planar* spaces shared between the input images. Our method uses such common rectified planes for inpainting a given region visible in several input images, and thus naturally

provides multi-view consistency, strongly encourages correct perspective and provides consistent depth completion. We fit planar segments to each 3D region to be completed and perform inpainting in a plane which can be seen as a fronto-parallel image of each region. Use of planes provides a common reference between images, allowing us to develop a clustering approach for efficient treatment of large scenes.

Use of intermediate rectified planar spaces is intuitive and appealing, since it involves locally inpainting in an undistorted image space. However this approach poses two difficult problems that we address with our method. First, we need to identify local planes and create planar regions which have two important properties for multi-view inpainting: the regions must be well oriented and have well-defined edges with respect to the underlying structure of the object being inpainting (e.g., wall, floor, sidewalk). This step needs to be very efficient, since our goal is to treat hundreds of images. Second, we need to carefully handle inpainting resolution and image resampling, since our algorithm operates in two distinct spaces: the input image and rectified planar space.

Our method addresses these challenges in two main steps. The first step is a new planar region extraction algorithm, that finds a small set of planes for multi-view inpainting and efficiently assigns input image pixels to planar regions. The second step uses an intermediate rectified planar space for multi-view inpainting. Our approach performs inpainting in the intermediate planes using a *cascade* of progressively larger resolutions based on constraints in each of the input images in the multi-view dataset. We carefully handle image resampling guided by 3D information in all steps. An example result is shown in Fig. 1.

In summary, our contributions are:

- An efficient planar region extraction method that facilitates multi-view inpainting for large datasets.
- A multi-view inpainting method using an intermediate rectified planar space and cascaded resolution. Our inpainting method matches resolution to that in the input images, uses high-quality resampling, structure-preserving initialization and a resolution-dependent distance metric. Together, these elements result in significant improvement in quality compared to previous methods.

Our method includes a clustering approach allowing us to handle large datasets. We present results of our method on indoors and outdoors scenes, and demonstrate significant improvement over previous work, especially in the context of IBR (Fig. 1(d), Fig. 13, 14, supplemental material and video).

2 PREVIOUS WORK

We briefly review the most closely related previous work in the domains of image inpainting, especially using 3D data for IBR.

2.1 Single Image Inpainting and Video Completion.

Inpainting is a vast research domain; a good survey can be found in Guillemot and Le Meur [2014]; we focus on the most closely related previous work.

The seminal work of Bertalmio et al. [2000] and Criminisi et al. [2004] have greatly influenced subsequent work in the fields of computer graphics and vision. Sun et al. [2005] use user inputs to better propagate structures during inpainting. More recently, the PatchMatch algorithm [Barnes et al. 2009] introduced efficient solutions for texture synthesis and inpainting. Several improvements have been proposed to the basic algorithm, including Image Melding [Darabi et al. 2012], that identifies and exploits transformations during matching, leading to improved quality. This term leads to better global appearance and less blurry results. He and Sun [2012] further exploit statistics of patch offsets to better guide inpainting. Recently deep neural networks (DNNs) and machine learning have been used for inpainting [Iizuka et al. 2017; Yang et al. 2017]. These methods combine global and local context information to achieve good quality results, but have limitations on image resolution and size of regions to complete. The lack of 3D information inherently limits single-image methods, which cannot enforce multi-view coherence, nor truly enforce perspective during inpainting; in contrast, we use multi-view 3D reconstruction to estimate planar structures, addressing these shortcomings.

Video completion is also an active field of research. The work of Wexler et al. [2007] introduced the methodological basis for many of the subsequent Expectation-Minimization methods, including ours. The recent video-based solution of Newson et al. [2015] proposes texture features which improve inpainting quality in many cases. Video-based methods have *dense, small baseline* sequences of frames with rich redundant information, in contrast to our sparse, wide-baseline capture for IBR.

2.2 Enhanced Inpainting with Structure and Multi-View Data.

Traditional single-image inpainting techniques are generally agnostic to the 3D content of the underlying scene, inducing errors in inpainting such as incorrect perspective or errors in planar structures. Previous methods [Huang et al. 2014; Sasao et al. 2016] use image analysis to find vanishing lines and induce approximate planar structure or perspective. The results are often impressive, but the lack of good quality 3D information limits their applicability at the scale required for IBR in complex scenes.

Depth information and multi-view data have been used to improve inpainting. The DCSH approach [Eshet et al. 2013] operates on RGBD images, while Howard et al. [2014] operate on stereo pairs, as opposed to our wide-baseline data. DCSH is based on a local planar approximation of the surface at each pixel that is sensitive to noise of depth images and is not applicable to missing geometry. Whyte et al. [2009] used several photographs of a scene, typically taken from the internet and simple registration between images to improve inpainting. Baek et al. [2016] jointly estimate depth and color on an image sequence, but do this progressively from one image to the next. The resulting depthmaps are thus not adapted to a free-viewpoint IBR context. Thonat et al. [2016] introduce the first method for multi-view inpainting with output suitable for free-viewpoint IBR. Their approach imposes soft multi-view coherence while inpainting separately in each input image. This soft optimization constraint does not provide true 3D coherence, and sometimes results in gradual changes in inpainted structures across an image sequence. The method also does not preserve perspective correctly, and has limitations in the quality of

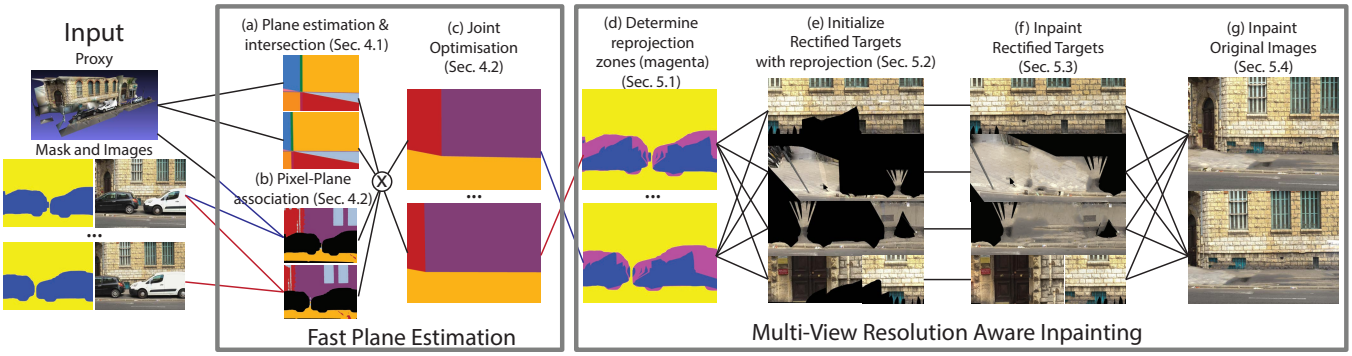

Figure 2: Overview of our method. We use a multi-view dataset and the corresponding 3D proxy as input. The first step is a fast algorithm to estimate local planes and assign them to pixels in the input images. In the second step we identify regions to inpaint, and create rectified targets with reprojection. We then inpaint the rectified target images and finally resample them to inpaint the original images.

inpainted depth since it is created by simple scan line interpolation. We address all these shortcomings in our solution by using locally common planar spaces, and show comparisons to several of these methods in Sec. 7.2.

2.3 Multiview Plane Estimation and Texture Mapping.

We develop a fast planar estimation approach to allow high-quality multi-view inpainting. Plane estimation is a central component of many 3D reconstruction algorithms, including for image-based rendering [Sinha et al. 2009]. Several methods [Bódis-Szomorú et al. 2014; Gallup et al. 2010] use Markov Random Field (MRF) solutions to estimate planes in a multi-view scene, often using higher-level structures. Sinha et al. [2008] introduce plane intersections to represent corners. We use standard methods to estimate the set of planes we will use; The novelty of our method lies in a fast algorithm to assign input image pixels to planes and preserve clean plane boundaries, facilitating high quality multi-view inpainting.

Finally, the idea of inpainting in rectified space has some similarities to texture mapping scenes captured with multi-view stereo (e.g., [Berjon et al. 2015; Callieri et al. 2008; Gal et al. 2010; Waechter et al. 2014; Zhou and Koltun 2014]). A recent approach [Bi et al. 2017] proposes a patch-based optimization for texture mapping from multiple images. Some of these methods show limited inpainting of small regions on object surfaces, but do not inpaint geometry. In very recent work, planar approximations are used for limited hole filling and high-quality texture mapping [Huang et al. 2017]. While we share methodological choices to these solutions (e.g., Poisson image editing for color harmonization or patch-based optimization [Bi et al. 2017]), the two problems are quite different: our final goal is to fill large unknown regions by inpainting the *input images* with resolution adapted to each input view, rather than mapping input image pixels to a mesh in a *view-independent* manner for texturing. For this reason, direct comparison to these methods is not possible.

3 OVERVIEW

Fig. 2 presents an overview of our method, with references to the corresponding sections in the text.

3.1 Input data

The input to our method is a set of images from different viewpoints of a scene containing man-made structures. These are typically photographs, taken with a reasonable "up" vector. We also require masks to identify the zones to inpaint; These can be obtained automatically using e.g., a CNN detector to find bounding boxes for cars, motorbikes or people [Gidaris and Komodakis 2015; Lin et al. 2017]. Alternatively, an interactive interface can be used to define objects to remove (see Sec. 7.1 and video). We use structure from motion (SfM) [Snavely et al. 2006] and multi-view stereo (MVS) [Jancosek and Pajdla 2011], to calibrate the input cameras and obtain an approximate 3D mesh or *proxy* of the scene, which is correctly scaled and stored in meters; Fig. 2, left. Our algorithm identifies planar regions in the images and operates in *rectified plane space*. We illustrate images, 3D reconstruction and rectified plane space in Fig. 3. We call the masked regions in the images *image targets*.

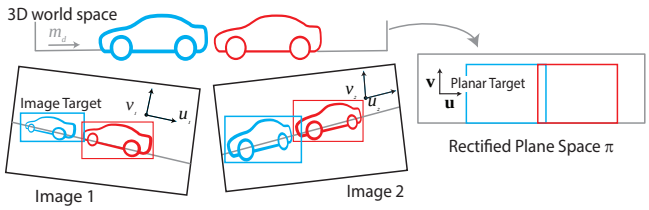

Figure 3: Our input is set of images and a multi-view stereo 3D reconstruction, together with the image targets, shown here as bounding boxes. We identify planar structures, e.g., the wall, and inpaint in the rectified plane (right); the inpainting result is then reprojected into the input images.

Our main goals are to provide good quality multi-view coherence, preservation of perspective during image inpainting and depth completion, overcoming the limitations of previous methods. We achieve this by using an intermediate planar space for inpainting. The first step is to identify locally planar structures in the scene and create rectified planes π for inpainting. *Image sources* are the pixels of the input images that are not contained in image targets, and

the corresponding pixels in the rectified plane are *planar sources*. We process images in clusters, allowing our method to scale to large datasets. Our approach correctly handles interdependencies between clusters, by reprojecting already inpainted regions into following clusters. For each cluster, our method has two main steps: plane estimation and inpainting.

3.2 Plane Estimation

Our main goals for the first step are speed, to allow treatment of large multi-view datasets and quality, especially to provide "clean edges" for corners between man-made locally planar structures. We introduce a fast method to identify planes in the scene and assign pixels of each input image to the corresponding planes; Fig. 2(a)-(c).

3.3 Multi-View Inpainting

In the second step, we use the intermediate rectified planar space to perform multi-view coherent inpainting that preserves perspective, and provides consistent inpainted depth using the planes; Fig. 2(d)-(g). Our approach exploits the multi-resolution nature of our algorithm to inpaint input images only at the required resolution, by introducing a *cascaded resolution* structure. Our inpainting method introduces a structure-aware initialization step and a resolution-dependent term, improving overall quality. Careful treatment is required for resampling and reprojection between the rectified planes and the input images. These are the central components of our efficient and coherent multi-view inpainting algorithm.

We show results of our approach on scenes containing up to hundreds of input images, both for indoors and outdoors scenes containing man-made structures. In Sections 4 and 5 we present our multi-view inpainting method for a single cluster which can be seen as an independent scene; we present our clustering method in Sec. 6.

4 FAST PLANE ESTIMATION FOR INPAINTING IN RECTIFIED PLANES

Good quality inpainting requires sharp plane boundaries and careful orientation of the planes, to avoid blur and "bleeding" artifacts between structures. We first estimate a small number of planes that serve as intermediate rectified spaces for multi-view inpainting. We then use our new fast approach to assign pixels to planes while respecting corners or equivalently *plane intersections*.

4.1 Estimation and Clustering of Planes

We start with a standard RANSAC plane estimation step, using the 3D points of the reconstruction, similar to previous work (e.g., [Bódis-Szomorú et al. 2014; Gallup et al. 2010]). We then perform fast hierarchical clustering based on the following distance between two planes π_1 and π_2:

$$d = (1 - \vec{n}_1 \cdot \vec{n}_1) + \chi_E(|d_1 - d_2|) \tag{1}$$

where $\vec{n}_i$ and d_i are respectively the normal and the distance of plane π_i to the origin, χ_E is the characteristic function of $E = \{x \in \mathbf{R}^+ | x < \alpha_\pi\}$, and α_π represents a maximum threshold for the variation of the distance to the origin between two planes. In our experiments α_π is set to 30cm in outdoors scenes and 10cm for indoors, which are reasonable thresholds to distinguish different objects for each scene category. The combination of plane estimation and clustering allows us to have a small number of planes while preserving good precision on plane position and orientation.

In the man-made scenes we target, structures are often locally parallel or perpendicular to intersections between planes, e.g., the ground and a wall or the corner of a building. This is similar to Manhattan world assumptions made in some 3D reconstruction algorithms [Furukawa et al. 2009; Huang et al. 2017]. Our goal is to orient the rectified planes to follow these directional structures; this simplifies the inpainting task, since the patch search and match steps become more reliable. We orient the planes, and define a local basis $\vec{\mathbf{u}}$, $\vec{\mathbf{v}}$ in the rectified plane, in world coordinates. Given that we focus on man-made structures, inpainting quality depends heavily on the orientation of these vectors. We first determine the best candidate for a "ground" plane, based on the assumption that photos are taken with a reasonable **up** vector. To do so we compute a normalized median vector over the **up** vectors of the input cameras. The ground is the plane whose normal has the highest dot product with this median vector. We can now compute a *main direction* vector $\vec{m_d}$ of the scene which we consider to be the intersection of the ground plane with a predominant vertical structure (e.g., a wall, see Fig. 3, top left). We search the set of planes with close-to-vertical orientation, and use the plane with the highest number of points to compute this intersection.

The basis of each plane is then obtained by reprojecting the main direction vector $\vec{m_d}$ on the plane. We use this projection as the first vector to build an orthonormal basis. If this projection norm is too small to be numerically stable, we use the ground normal instead.

The two vectors of the oriented bases are thus:

$$\vec{\mathbf{u}} = \frac{\vec{m_d} - (\vec{n} \cdot \vec{m_d})\vec{n}}{|\vec{m_d} - (\vec{n} \cdot \vec{m_d})\vec{n}|} \text{ and } \vec{\mathbf{v}} = \frac{\vec{n} \times \vec{\mathbf{u}}}{|\vec{n} \times \vec{\mathbf{u}}|} \tag{2}$$

We now have the main planar structures of the scene and basis for each plane. The basis vectors can be projected into each image I_i; we denote the image space basis vectors as $\vec{u}_i$ and $\vec{v}_i$. These are defined as follows, for a pixel (x, y):

$$\vec{u}(x, y) \;=\; C(P(x, y) + \vec{\mathbf{u}}) - (x, y) \tag{3}$$

where $P(x, y)$ is the 3D point in the plane corresponding to the pixel (x, y) and C is the projection operator for this camera. These three spaces are illustrated in Fig. 3.

4.2 Assigning Pixels to Planes

We now need to assign pixels to planes to create the intermediate rectified planes for multi-view coherent inpainting. If we compute the 3D position of each source pixel using the camera pose and approximate 3D proxy, and then associate the pixel to its closest plane, we have noisy results in several regions, and in particular at plane boundaries, see Fig. 4. For good quality inpainting, it is essential to have clean boundaries, avoiding content being mixed between distinct surfaces.

We observe that if we keep a small number of planes, these subdivide the input images into a limited number of zones K, where $K \leq 50$ in our tests. The labelling problem of assigning each zone to a plane can thus be solved efficiently, and we can use the accurate plane intersections for clean boundaries. Plane intersections have

Figure 4: Left: original image. Right: pixel-plane association.

been used in the different context of image-based modelling [Sinha et al. 2008]; our solution benefits from the quality of the clean intersection edges, and provides a very efficient solution, avoiding the need for more expensive pixel-based MRF methods [Bódis-Szomorú et al. 2014; Gallup et al. 2010; Sinha et al. 2008].

We compute the intersections between all clustered planes in 3D and reproject these intersection lines into the different images to inpaint. For efficiency, we use a bitwise encoding: for each re-projected intersection line, each pixel receives a 0 or 1 code if it is on the left or the right side of the line respectively. This bitwise code separates the images into zones, and two zones are connected if and only if their bitwise code differs by one bit. We can see the intersections and zones in Fig. 5.

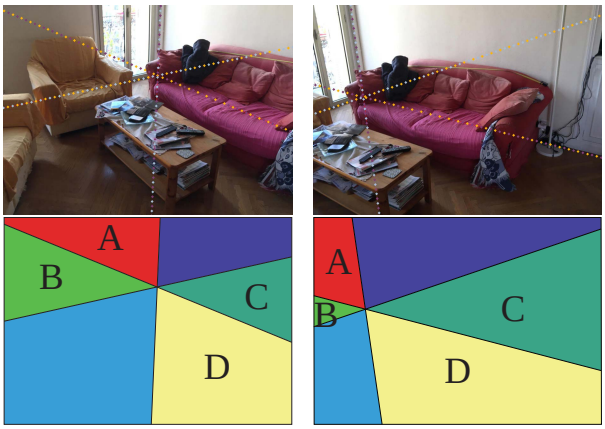

Figure 5: Two images of the living room dataset. We see that several zone intersections (e.g., of the zones A and B) are incorrect and need to be removed.

Each image is now separated into zones and we want to associate a plane to each one of them. To do this we define an energy function. We have the set of zones $Z = \{z_0, z_2 \dots z_n\}$, the set of planes Π and a given plane labeling $L(z_i) \in \Pi$. We first introduce a compactness constraint given by the pixel-plane association for source pixels. If a zone has a majority of pixels associated to one plane we want to encourage the association of the entire zone to this plane. We express this with the following term:

$$e_{\mathbf{z}}(z_i) = \max_{\pi \in \Pi} P_{\mathbf{N}}(z_i, \pi) - P_{\mathbf{N}}(z_i, L(z_i)) \tag{4}$$

where $P_{\mathbf{N}}(z, \pi)$ is the number of pixels associated to plane π or any plane parallel to π in zone z. This term implicitly treats visibility, since closer points tend to cover a larger area.

We also encourage zones to be consistent with the plane intersections, while still being able to discard intersections in some cases, e.g., when we have occlusions between planes, the intersections are present both on the visible and occluded side. In Fig. 5 the plane intersection line that separates A,B and C,D is relevant only for C and D and should be discarded for A and B. This is expressed with the term:

$$e_c(z_i, l_k, z_j, l_m) = \begin{cases} 0, & \textbf{if } l_k \in C(z_i, z_j) \text{ and } l_m \in C(z_i, z_j) \\ \infty, & \textbf{otherwise} \end{cases} \tag{5}$$

where $C(z_i, z_j)$ is the set of the two planes that intersect, forming the intersection line connecting z_i and z_j. Note that label l_k may be equal to l_m in the case where e_c is 0. In Fig. 5, this term encourages points in both zones A and B to be assigned the cyan plane (label), while encouraging points in D and C to be assigned to the orange and magenta label respectively, using the color coding of Fig. 4 and 6. The final energy is thus:

$$E_z = \sum_{z_i \in Z} e_{\mathbf{z}}(z_i, L(z_i)) + \sum_{\{z_i, z_j\} \in Z_c} e_c(z_i, L(z_i), z_j, L(z_j)) \tag{6}$$

where $Z_c = \{\{z_i, z_j\} \mid z_i \text{ and } z_j \text{ are connected }\}$. Since the number of zones is small, less than 100 in all our examples, we use a greedy search in the tree of solutions to optimize. In practice this step is very fast, taking less than 50 ms in all our datasets. Once complete, we have an approximate planar representation of the scene suitable for high-quality rectified inpainting. Results are shown in Fig. 6.

Figure 6: The result of our fast pixel plane assignment step. Incorrect zones have been removed.

5 MULTI-VIEW, RESOLUTION-AWARE INPAINTING

We now have a set of planes associated to a set of images and we can proceed with rectified multi-view inpainting in each plane. A major advantage of having estimated planes is that we have synthesized 3D positions for all target pixels. These 3D positions allow us to identify target regions that are connected between the images, for example corresponding to the same object seen from different viewpoints. Inpainting is performed in three steps: 1) Creation of the resolution-aware rectified images with target

regions, 2) Reprojection to create source pixels for inpainting and 3) Rectified multi-view inpainting.

5.1 Creating Resolution-aware Rectified Images for Inpainting

Inpainting in rectified space can be wasteful if done naively, since the final goal is to inpaint the *input* images, which have a given resolution for each image target. Consider the car in Fig. 8: due to strong perspective each part of the car occupies a progressively smaller region in the input image I_1. Inpainting at high resolution everywhere in the rectified plane would be wasteful, both in computation and storage of the inpainted images.

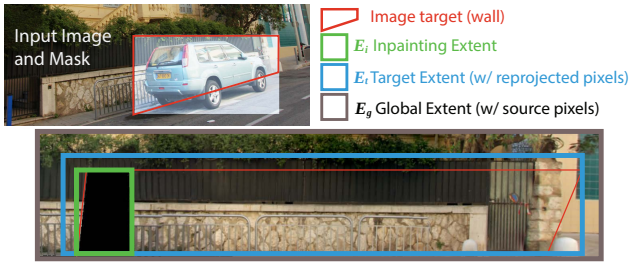

Figure 7: The different extents. The red region (wall) is projected into the plane and used to define the three extents: pixels to inpaint E_i, target E_t including reprojected pixels and E_g including the source pixels.

We distinguish three *extents* in each rectified plane π. The inpainting extent E_i, i.e., pixels $\mathcal{P}_i$ in the planar target, corresponding to 3D points in π that are not covered by a source pixel in any input image. We also have the remaining pixels $\mathcal{P}_r$ of the planar target, that can be filled with pixels reprojected from input images. The *target extent* E_t contains E_i and pixels $\mathcal{P}_r$. Finally, we have pixels $\mathcal{P}_s$ taken directly from image sources as *source* pixels for inpainting: E_t and pixels $\mathcal{P}_s$ define the global extent E_g of the rectified planar image to be inpainted. Example extents are shown in Fig. 7. We first explain how we obtain E_i, E_t and E_g and then how we build a *cascade* of different resolutions.

5.1.1 Creating the Inpainting and Target Extents. We first segment each image target into regions corresponding to each plane; e.g., in Fig. 7 we split into a region for the wall (shown in red) and for the ground. We determine the pixels to be inpainted for this region in image I_i by reprojecting the pixels into all other images using the corresponding 3D position in the plane. If a point projects outside an image target in another image I_j, the pixel in I_i can be filled by reprojection and belongs to set $\mathcal{P}_r$. Otherwise the pixel is marked to be inpainted and is in $\mathcal{P}_i$. We also check for overlap between all pairs of image inpainting regions in different images. If there is more than 20% overlap, we mark these regions as a group. This ensures that we do not merge regions that barely overlap, and account for masks inaccuracy.

For each group of regions to inpaint, we project pixels to inpaint $\mathcal{P}_i$ into the rectified plane, and find the corresponding bounding box which defines the inpainting extent E_i. We then associate each pixel $\mathcal{P}_r$ to the closest inpainting extent. The bounding box of the union of E_i and associated pixels $\mathcal{P}_r$ is the target extent E_t that is clipped to the boundaries of the reprojected input images (Fig. 7).

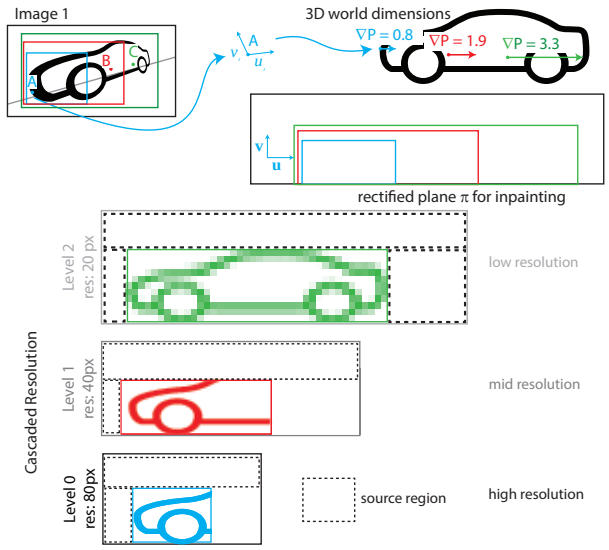

Figure 8: Cascaded Resolution: the maximum resolution required is found using the gradient of the rectified basis vectors u an v.

5.1.2 Cascaded Resolution for Rectified Inpainting. Our inpainting approach is based on PatchMatch [Barnes et al. 2009] which is inherently multi-resolution: inpainting starts at a coarse resolution, followed by upscaling and inpainting at progressively higher resolutions. Since we will adapt resolution to the regions of the input images, we create a *cascade* of rectified planar images, each of which is inpainted only at the required resolution: e.g., in the example of Fig. 8, only Level 0 (black) is inpainted at the highest resolution, Level 1 (mid grey) at half resolution and Level 2 (light gray) at the coarsest resolution.

To do this, for each pixel in E_t we determine the required resolution using an approach similar to a mipmap lookup in texture mapping [Heckbert 1989], using the directional derivatives $\nabla\vec{u}(P(x,y))$ and $\nabla\vec{v}(P(x,y))$. We provide details in supplemental material. The specific resolution of each rectified image will be determined based on the value of ∇P and the resolution of the input images. We first determine the maximum resolution required (blue region in Fig. 8), and we then create n levels for inpainting, by halving the maximum resolution, corresponding to the minimum value $P_{\min} = \min\nabla P$. In the example of Fig. 8, level 0 corresponds to maximum resolution, and will include pixels with ∇P between $P_{\min}$ and 2 $P_{\min}$. Pixels with ∇P between $P_{\min}$ and 4 $P_{\min}$ will be in level 1 etc. (see Fig. 8). At the end of this process we perform a region expansion step, and find the region including only level 0 pixels, then the region with level 1, then level 2 etc. resulting in the cascaded resolution structure shown in Fig. 8. We ensure that E_t is fully contained in E_g, so that all reprojected pixels are present in the source. We take 50% of the inpainting region size on each side as source pixels $\mathcal{P}_s$. If there is not enough source on one side to reach 50% we increase

the amount of source on the other side correspondingly. Example source regions are shown as dashed boxes in Fig. 8.

5.2 Resampling and Reprojection

We now have the mapping from input images to the target planar regions. Given the projective transformation between the input images and the rectified plane, we need to carefully resample the input images to provide good quality source pixels for inpainting. Each pixel in a rectified image is backprojected into the original images, using Elliptical Weighted Average (EWA) filtering [Heckbert 1989] to sample the input images and provide source colors. We call valid rectified pixels those with coordinates within E_g and that are not target pixels. For input image pixels that re-project on valid rectified pixels in more than one image we have to select which image to sample. We formulate this as a labeling problem. We want each pixel to be sampled with a high quality kernel, but also from pixels that correspond to 3D content close to the plane. We thus introduce the following quality term for a pixel p sampled in image I^l, with label l:

$$D_p(I^l) = A_p(I^l) + \gamma D_M(I^l), \tag{7}$$

where $A_p(I^l)$ is the inverse of the area of the EWA filter to fill pixel p coming from image I^l [Heckbert 1989]. D_M is the distance between the point given by the plane equation and the 3D position obtained with the original depth image associated to I^l, given by the proxy. The first term favors sampling images with large elliptical kernels, which means that we have more accuracy on the sampled values while the second term favors pixels corresponding to points in the 3D reconstruction closer to plane (see Fig. 9). The factor γ is used to balance the two terms. Ideally $A_p(I^p)$ should be 1 or less, i.e., the sampling kernel for one pixel in rectified space has a surface of one pixel in the image. Since our scenes are in meters, we set $\gamma = 10$ so that a few centimeters deviation from the plane does not have a big impact on the term in Eq. 7.

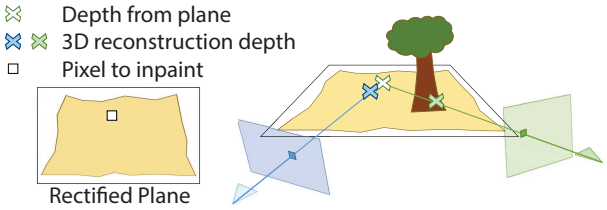

Figure 9: The Unary term D_M encourages the use of the depth from the images with depth closer to the plane.

For each pixel we find l which minimizes $D_p(I^l)$. If we only had $A_p(I^l)$, this per pixel minima would lead to homogeneous source label regions. However the noise in the original proxy breaks this regularity through $D_M(I^l)$, leading to noise in label selection at boundary of low energy zones. We overcome this issue by solving a multi label MRF only for pixels that do not have homogeneous minima in a 10×10 neighborhood. This occurs for a small number of pixels – less than 10% of the image pixels in our tests – limiting the impact on computation time compared to solving the MRF on the entire image. Similar steps exist in previous work [Thonat et al. 2016; Whyte et al. 2009], but in contrast to those approaches we have well-defined geometry, allowing us to define the following energy:

$$E_s = \sum_{p \in R_{im}} D_p(I^l) + \sum_{\{p,q\} \in N} V_{p,q}(I^l, I^k) \tag{8}$$

where V is a standard smoothness term (see supplemental material), that helps borders of the zones to be coherent in terms of visual content, and N represents a 4-neighborhood.

The result of the graph-cut and the local minima of the homogenous zones provides the source for each rectified pixel. We sample these sources by reprojecting the input images into the rectified image. Because pixels coming from different images may have illumination variations we run a Poisson blending step [Pérez et al. 2003] at the border defined by source changes. A planar source and its source labels are shown in Fig. 10.

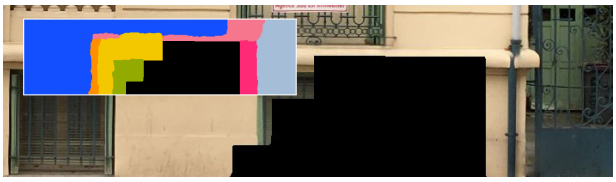

Figure 10: Rectified plane initialization. Inset: each color label corresponds to a different input source image.

5.3 Inpainting rectified images

For each global extent corresponding to a rectified planar region, we now have the cascaded resolution structure and we can perform inpainting. We use an Expectation-Maximization algorithm similar to previous work ([Barnes et al. 2009; Thonat et al. 2016; Wexler et al. 2007]): we first initialize then proceed with several iterations of PatchMatch [Barnes et al. 2009] followed by voting.

5.3.1 Initialization. To initialize our images at the lowest scale, we strive to preserve major structures. We represent these structures by strong gradients in the images, and encourage the initialization step to preserve these boundaries.

We first compute straight lines in all filled areas of the image at the highest resolution using a Canny edge detector and a Hough transform. The total number of lines found is N_L. At the lowest resolution, we associate each target pixel p_t with coordinates t to a source pixel p_s with coordinates s randomly sampled according to the following distribution, that discourages sampling pairs (s,t) crossing many lines and thus respects structures in the sources. The sampling density associating a target t to a source s is defined as:

$$p(p_t \leftrightarrow p_s) = \frac{1}{Z_t} G(s,t) \quad \text{and} \quad Z_t = \sum_{s \in S} G(s,t) \tag{9}$$

with

$$G(s,t) = e^{-0.5 \frac{\|t-s\|_2^2}{((w+h)/2)^2}} e^{-0.5 (\frac{N_l(p_t,p_s)}{\sigma_l})^2}$$

where w, h are the width and height of the rectified image at lowest resolution, $N_l(X_t, X_s)$ is the number of lines intersecting (t,s) and $\sigma_l = 0.05\ N_L$.

We finally initialize each pixel in the target area E_i by transferring the patch at the associated source area to the target patch and performing mean blending of all patches per pixel.

5.3.2 Inpainting. We use the following distance between patches t (target) and s (source) for both the PatchMatch step and voting:

$$d(t,s) = \|t-s\|_2^2 + \lambda_{\text{occ}} \frac{\Omega(s)}{\omega_{\text{best}}} + \lambda_{\text{tfeat}} T(t,s) + \lambda_{\text{res}} R(t,s) \quad (10)$$

where $\frac{\Omega(s_i)}{\omega_{\text{best}}}$ is a spatial uniformity ("occurence") term described by Kaspar et al. [2015] and $T(t_i, s_i)$ is the texture feature descriptor distance from Newson et al. [2015]. We found that images with many structured features need high uniformity to prevent blur, while imposing uniformity when in less structured regions leads to inconsistent copies of blocks of content (e.g., a piece of one structure in the middle of another).

Instead of setting λ_{occ} manually as in Kaspar et al. [2015], we automatically choose λ_{occ} with respect to the mean texture feature norm:

$$\lambda_{\text{occ}} = 0.01(\frac{1}{|S|}\sum_{s\in S} \|Tf(s)\|_2)^2 \quad (11)$$

We introduce the term $R(t,s)$ that is a measure of the correspondence of the original resolution between two pixels in the source image. To compute R we reason on rectified source pixels p_r in $\mathcal{P}_s$ and the corresponding pixels p_s in the input images. For each pixel p_r in the initialized rectified image we have the corresponding original input image pixel p_s, and the elliptical sampling kernel e_s used to sample p_s. Recall that $A_p = \frac{1}{\text{Area}(e_s)}$. For a target pixel we compute A_p for all the input images in which it reprojects and we take the one with the largest area $A_t(I_{\min}(t))$:

$$R(t,s) = \max(0, A_s(I_s) - A_t(I_{\min}(t))) \quad (12)$$

given:

$$I_{\min}(t) = \underset{I\in\mathcal{I}}{\text{argmin}} \|A_t(I)\| \quad (13)$$

where $\mathcal{I}$ is the set of all input images.

This ensures coherent resolution of the patches used to inpaint a specific zone, and also provides sufficient resolution for all the images when we reproject back into the original image space. λ_{res} is set to 0.1 and λ_{tfeat} to 0.001 for all our tests. The different maps for resolution, texture feature and uniformity are shown in supplemental material.

5.3.3 Proxy and Depth synthesis. IBR algorithms require a coherent geometric proxy. Since we have removed objects from the scene, the original reconstructed proxy cannot be used. We first create a clean version of the mesh by removing all vertices corresponding to pixels contained in target regions in all images. We do this by projecting all vertices of the mesh into the input images; if a vertex reprojects in a target region in one image it is marked as potentially invalid and if it reprojects into a source region in one image it is marked as visible. Vertices are that are potentially invalid and not visible are marked as invalid. The clean mesh consists of the triangles that have three valid vertices, and contains holes in regions corresponding to the E_i of the input images.

We fill these holes using the planes of the corresponding rectified images. We create a mesh by inserting a vertex every 10 pixels in the rectified image.

5.4 Inpainting input images

After inpainting the rectified planar image, we inpaint the input images by sampling the plane using the same resampling and reprojection approach as for the creation of E_t (Sec. 5.2). We compute the elliptical kernels but this time from rectified space to image space. We then apply Poisson blending on the border of the zones to inpaint, to compensate for view-dependent color differences. As a final step we propagate structures that do no not belong to any estimated plane when they are in a source region in at least one image. Specifically, if an inpainted pixel projects on a source pixel in a nearby image with original proxy-based depth closer than the depth of the plane by at least α_π, we sample this source pixel directly.

6 HANDLING LARGE DATASETS

We treat large datasets using a clustering step and reprojection between overlapping clusters. We set a target size of a cluster N_C, depending on the available memory and computing resources. We first compute the clean version of the proxy; this provides a common reference so we can we can treat each cluster separately and thus avoid loading all images in memory at the same time. Next we cluster input cameras with k-means (k = N_C) by 3D position and angle with two-thirds/one-third weights respectively. This creates a set of independent clusters which can be seen as separate scenes. We start by inpainting one of these sets. After treating a set, we reproject the rectified inpainted images of all treated sets into each image of the next set. If a rectified inpainted image reprojects into zones to inpaint, the pixels are sampled in the corresponding rectified image and considered inpainted. For the rectified plane initialization step we add in the original source images, i.e., all images whose cameras see the polygons defined by the rectified images. This maintains and propagates multi-view coherence across sets and additionally avoids inpainting the same 3D zone twice while taking into account all the available information.

7 IMPLEMENTATION, RESULTS AND COMPARISONS

We implemented our method in C++, parallelizing over images and planar targets when possible. All running times are on a PC with a dual Intel Xeon E5-2650 CPU and 64Gb of memory. Plane processing and inpainting are run offline, and create the data that can be used in our IBR system. All IBR results are with a per-pixel Unstructured Lumigraph [Buehler et al. 2001] implementation with soft visibility [Eisemann et al. 2008].

7.1 Results

We show results for outdoors and indoors scenes. All running time statistics are shown in Table 1. In Fig. 1, 11, we show a Street scene with 290 images, using masks automatically computed with RefineNet [Lin et al. 2017]. In row 3 of Fig. 11 we show pixels filled by reprojection in magenta and by inpainting in blue, illustrating how our method successfully inpaints very large image regions. We also show result for another Street scene from [Thonat et al. 2016] (Street 2); we show results for 3 additional scenes in supplemental material and the accompanying video. In Fig. 12, we show a living room scene where the sofa was removed using a interactive tool:

Figure 11: Results from the Large Street dataset for 4 images. First row: input images. Second row: inpainting result. Third row: inpainting result with overlay for reprojected regions (magenta) and inpainted regions (blue). Notice how perspective effects on the sidewalk tiles and the separation between structures (wall, sidewalk) are preserved during inpainting.

Figure 12: Results for the living room dataset.

Scene	N	Res	N_C	T_p	T_i
Fig. 13	16	2016x1512	na	1'49"	2'52"
Living Room	40	1517x1110	na	16'50"	8'46"
Large Street	290	1828x984	70	45'	28'
Fig. 15	30	2546x1672	na	5'25"	7'40"

Table 1: For each scene we list the number N of input images, their resolution *Res*, the cluster value N_C where applicable, the time T_p for plane processing and T_i for inpainting in minutes.

the object is removed with one selection in the proxy using a simple 3D viewer such as Meshlab [Cignoni et al. 2008] (please see video) and the masks are automatically created in all input images through reprojection. Please see the accompanying video which shows free-viewpoint IBR on all scenes, before and after inpainting.

7.2 Comparisons

All comparisons shown were generated either using the original authors' code under their guidance ([Yang et al. 2017]) or by the respective authors themselves (all others). We show comparisons to single image methods in Fig. 13. Evidently, our approach has an advantage over these solutions since we have use multiple input images; the main goal of this comparison is to demonstrate that such methods are not well suited to our goal of multi-view inpainting for IBR.

We see that our approach preserves perspective and reduces blur significantly compares to these methods. In Fig. 14, we show comparisons to methods that treat multi-view datasets, i.e., Whyte et al. [2009], Thonat et al. [2016] and Baek et al. [2016]. We see that our approach overcomes the artifacts related to perspective by better preserving slanted lines (e.g., on the sidewalk) and greatly reduces blur. Our approach also achieves better overall multi-view consistency, e.g., the sidewalk and road in scenes from Thonat et al. [2016]. In supplemental material, we show Fig. 13 and 14 without insets for closer inspection, additional comparisons and more examples of improvement for multi-view consistency.

7.3 Limitations

In Fig. 15 we show inpainting on a non-planar surface. The method performs reasonably well overall, but non planar structures are incorrectly copied to a plane, e.g., the curved stairs copied onto the wall.

Another limitation of our approach is related to the number of planes we find during plane estimation. For example, in the Living Room scene, we do not identify the plane of the coffee table with

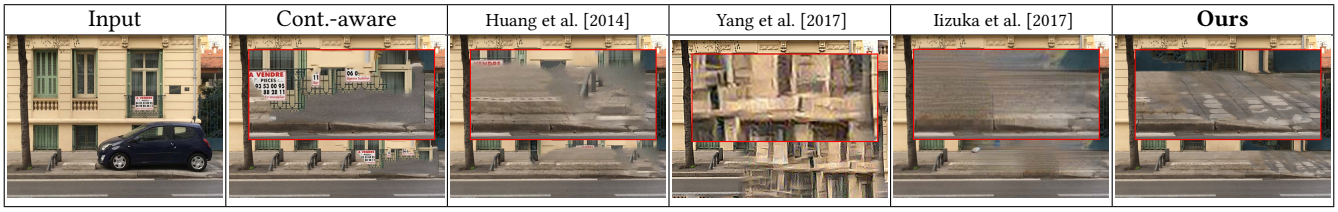

Figure 13: Comparison to single image methods. Cont.-aware is the result of Content-Aware fill in Photoshop CC (2016 edition); note that the method of Yang et al. [2017] operates on much lower resolution images.

Figure 14: Comparison to multi-view methods.

the default parameter settings. This is a user tunable parameter, and can be adapted depending on the application needs.

Figure 15: Inpainting on a curved surface.

8 DISCUSSION AND CONCLUSIONS

We have presented a scalable inpainting method for multi-view datasets, suitable for large-scale IBR, which greatly improves quality compared to previous work. We first present a fast approach to identify planes for inpainting, and assign input image pixels to these planes. Our algorithm enforces clean boundaries between planar regions, encouraging high-quality inpainting. Our approach carefully handles sampling and resolution between the input image and rectified planar spaces, by introducing a cascaded resolution structure used for inpainting. Our inpainting method introduces structure-aware initialization and a resolution term improving overall quality. We also present a clustering approach to handle large datasets. We demonstrated our results on indoors and outdoors scenes with up to hundreds of input images. Our results show significant overall improvement in inpainting quality, providing a usable solution for manipulating IBR environments.

In future work, we envisage the use of semantic information to improve the selection of sources for inpainting, thus further improving final image quality overall. Our clustering approach can be extended to provide good quality source content even when there is no overlap between sets; semantic information will probably be required here as well to ensure good quality transfer of sources between such disjoint clusters.

ACKNOWLEDGMENTS

This research was partially funded by the European Union Horizon 2020 research and innovation programme under grant agreement No. 727188 "EMOTIVE", http://www.emotiveproject.eu/, the French research grant ANR SEMAPOLIS and by generous donations from Technicolor and Adobe. The authors thank T. Thonat for his help and S. Bonopera for system support.

REFERENCES

Seung-Hwan Baek, Inchang Choi, and Min H. Kim. 2016. Multiview Image Completion with Space Structure Propagation. In *Proc. of IEEE Conference Computer Vision and*

Pattern Recognition (CVPR). IEEE, Las Vegas, USA.

Connelly Barnes, Eli Shechtman, Adam Finkelstein, and Dan B Goldman. 2009. PatchMatch: A Randomized Correspondence Algorithm for Structural Image Editing. *ACM Transactions on Graphics* 28, 3 (Aug. 2009).

D Berjon, F Moran, N Garcia, et al. 2015. Seamless, Static Multi-Texturing of 3D Meshes. In *Computer Graphics Forum*, Vol. 34. Wiley Online Library, 228–238.

Marcelo Bertalmio, Guillermo Sapiro, Vincent Caselles, and Coloma Ballester. 2000. Image inpainting. In *Proc. SIGGRAPH*. 417–424.

Sai Bi, Nima Khademi Kalantari, and Ravi Ramamoorthi. 2017. Patch-based optimization for image-based texture mapping. *ACM Transactions on Graphics* 36, 4 (2017).

András Bódis-Szomorú, Hayko Riemenschneider, and Luc Van Gool. 2014. Fast, approximate piecewise-planar modeling based on sparse structure-from-motion and superpixels. In *Proc. of IEEE Conference on Computer Vision and Pattern Recognition(CVPR)*. 469–476.

Chris Buehler, Michael Bosse, Leonard McMillan, and Steven Gortlerand Michael Cohen. 2001. Unstructured lumigraph rendering. In *Proc. SIGGRAPH*.

Marco Callieri, Paolo Cignoni, Massimiliano Corsini, and Roberto Scopigno. 2008. Masked photo blending: Mapping dense photographic data set on high-resolution sampled 3D models. *Computers & Graphics* 32, 4 (2008), 464–473.

Rodrigo Ortiz Cayon, Abdelaziz Djelouah, and George Drettakis. 2015. A bayesian approach for selective image-based rendering using superpixels. In *3D Vision (3DV), 2015 International Conference on*. IEEE, 469–477.

Paolo Cignoni, Marco Callieri, Massimiliano Corsini, Matteo Dellepiane, Fabio Ganovelli, and Guido Ranzuglia. 2008. MeshLab: an Open-Source Mesh Processing Tool. In *Eurographics Italian Chapter Conference*, Vittorio Scarano, Rosario De Chiara, and Ugo Erra (Eds.). Eurographics.

Antonio Criminisi, Patrick Pérez, and Kentaro Toyama. 2004. Region filling and object removal by exemplar-based image inpainting. *IEEE Transactions on image processing* 13, 9 (2004), 1200–1212.

Soheil Darabi, Eli Shechtman, Connelly Barnes, Dan B. Goldman, and Pradeep Sen. 2012. Image Melding: Combining Inconsistent Images Using Patch-based Synthesis. *ACM Trans. Graph.* 31, 4 (July 2012), 82:1–82:10.

Martin Eisemann, Bert De Decker, Marcus Magnor, Philippe Bekaert, Edilson de Aguiar, Naveed Ahmed, Christian Theobalt, and Anita Sellent. 2008. Floating Textures. *Computer Graphics Forum* (2008).

Yaron Eshet, Simon Korman, Eyal Ofek, and Shai Avidan. 2013. DCSH - Matching Patches in RGBD Images. In *IEEE International Conference on Computer Vision, ICCV 2013, Sydney, Australia, December 1-8, 2013*. 89–96.

Yasutaka Furukawa, Brian Curless, Steven M Seitz, and Richard Szeliski. 2009. Manhattan-world stereo. In *Proc. of IEEE Conference Computer Vision and Pattern Recognition (CVPR)*. IEEE, 1422–1429.

Ran Gal, Yonatan Wexler, Eyal Ofek, Hugues Hoppe, and Daniel Cohen-Or. 2010. Seamless montage for texturing models. In *Computer Graphics Forum*, Vol. 29. 479–486.

David Gallup, Jan-Michael Frahm, and Marc Pollefeys. 2010. Piecewise planar and non-planar stereo for urban scene reconstruction. In *Proc. of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 1418–1425.

Spyros Gidaris and Nikos Komodakis. 2015. Object detection via a multi-region & semantic segmentation-aware CNN model. *CoRR* abs/1505.01749 (2015).

Christine Guillemot and Olivier Le Meur. 2014. Image inpainting: Overview and recent advances. *IEEE signal processing magazine* 31, 1 (2014), 127–144.

Kaiming He and Jian Sun. 2012. Statistics of patch offsets for image completion. In *Computer Vision–ECCV 2012*. Springer, 16–29.

Paul S. Heckbert. 1989. *Fundamentals of Texture Mapping and Image Warping*. Technical Report UCB/CSD-89-516. University of California, Berkeley. http://www2.eecs.berkeley.edu/Pubs/TechRpts/1989/CSD-89-516.pdf.

Peter Hedman, Tobias Ritschel, George Drettakis, and Gabriel Brostow. 2016. Scalable inside-out image-based rendering. *ACM Transactions on Graphics* 35, 6 (2016), 231.

Joel Howard, Bryan S. Morse, Scott Cohen, and Brian L. Price. 2014. Depth-based patch scaling for content-aware stereo image completion.. In *WACV*. IEEE Computer Society, 9–16.

Jingwei Huang, Angela Dai, Leonidas Guibas, and Matthias Nießner. 2017. 3DLite: Towards Commodity 3D Scanning for Content Creation. *ACM Transactions on Graphics* 2017 (2017).

Jia-Bin Huang, Sing Bing Kang, Narendra Ahuja, and Johannes Kopf. 2014. Image Completion Using Planar Structure Guidance. *ACM Trans. Graph.* 33, 4 (July 2014), 129:1–129:10.

Satoshi Iizuka, Edgar Simo-Serra, and Hiroshi Ishikawa. 2017. Globally and locally consistent image completion. *ACM Transactions on Graphics (TOG)* 36, 4 (2017), 107.

Michal Jancosek and Tomás Pajdla. 2011. Multi-view reconstruction preserving weakly-supported surfaces. In *Proc. of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 3121–3128.

Alexandre Kaspar, Boris Neubert, Dani Lischinski, Mark Pauly, and Johannes Kopf. 2015. Self Tuning Texture Optimization. *Computer Graphics Forum* (2015). https://doi.org/10.1111/cgf.12565

G. Lin, A. Milan, C. Shen, and I. Reid. 2017. RefineNet: Multi-Path Refinement Networks for High-Resolution Semantic Segmentation. In *Proc. of IEEE Conference Computer Vision and Pattern Recognition (CVPR)*.

Alasdair Newson, Andrés Almansa, Matthieu Fradet, Yann Gousseau, and Patrick Pérez. 2015. Video Inpainting of Complex Scenes. *CoRR* abs/1503.05528 (2015).

Eric Penner and Li Zhang. 2017. Soft 3D Reconstruction for View Synthesis. *ACM Transactions on Graphics* 36 (2017).

Patrick Pérez, Michel Gangnet, and Andrew Blake. 2003. Poisson Image Editing. *ACM Trans. Graph.* 22, 3 (July 2003), 313–318.

Hiroto Sasao, Norihiko Kawai, Tomokazu Sato, and Naokazu Yokoya. 2016. A study on effect of automatic perspective correction on exemplar-based image inpainting. *ITE Transactions on Media Technology and Applications* 4, 1 (2016), 21–32.

Sudipta N. Sinha, Drew Steedly, and Richard Szeliski. 2009. Piecewise planar stereo for image-based rendering. In *IEEE ICCV 2009*. 1881–1888.

Sudipta N Sinha, Drew Steedly, Richard Szeliski, Maneesh Agrawala, and Marc Pollefeys. 2008. Interactive 3D architectural modeling from unordered photo collections. In *ACM Transactions on Graphics (TOG)*, Vol. 27. ACM, 159.

Noah Snavely, Steven M. Seitz, and Richard Szeliski. 2006. Photo tourism: exploring photo collections in 3D. *ACM Transactions on Graphics* (2006).

Jian Sun, Lu Yuan, Jiaya Jia, and Heung-Yeung Shum. 2005. Image completion with structure propagation. In *ACM Transactions on Graphics (ToG)*, Vol. 24. ACM, 861–868.

Theo Thonat, Eli Shechtman, Sylvain Paris, and George Drettakis. 2016. Multi-View Inpainting for Image-Based Scene Editing and Rendering. In *Fourth International Conference on 3D Vision, 3DV 2016*. 351–359.

Michael Waechter, Nils Moehrle, and Michael Goesele. 2014. Let There Be Color! Large-Scale Texturing of 3D Reconstructions. In *ECCV 2014*. Springer International Publishing, 836–850.

Yonatan Wexler, Eli Shechtman, and Michal Irani. 2007. Space-time completion of video. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 29, 3 (2007), 463–476.

Oliver Whyte, Josef Sivic, and Andrew Zisserman. 2009. Get Out of my PictureInternet-based Inpainting.. In *BMVC*. British Machine Vision Association.

Chao Yang, Xin Lu, Zhe Lin, Eli Shechtman, Oliver Wang, and Hao Li. 2017. High-Resolution Image Inpainting using Multi-Scale Neural Patch Synthesis. *ACM Transactions on Graphics* 2017 (2017).

Qian-Yi Zhou and Vladlen Koltun. 2014. Color map optimization for 3D reconstruction with consumer depth cameras. *ACM Transactions on Graphics* 33, 4 (2014), 155.